FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Gestão Centralizada de Parques Informáticos

Rodrigo Miguel Pereira Oliveira

Licenciado em Engenharia Electrotécnica e de Computadores
pela Faculdade de Engenharia da Universidade do Porto

Dissertação submetida para satisfação
dos requisitos do grau de mestre
em
Redes e Serviços de Comunicação

Dissertação realizada sob a orientação de
Mestre João Isidro Vila Verde

Assistente Convidado
do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

Porto, Dezembro de 2005

Agradecimentos

Gostaria de agradecer e dedicar este meu trabalho a uma série de pessoas que sempre me apoiaram e me transmitiram a força necessária para concluir esta missão.

Em primeiro lugar, um agradecimento particular ao meu orientador, Eng.º Isidro Vila Verde. Foi dele que partiu a ideia para o tema desta dissertação, o qual viria a revelar-se de uma extrema utilidade para a área profissional em que desenvolvo o meu trabalho. Muito obrigado por todo o empenho, acompanhamento, partilha de experiências e toda a ajuda que me forneceu ao longo deste período de trabalho. A sua dedicação ficará para sempre na minha memória.

Gostaria de deixar um agradecimento a todos os colegas com quem trabalhei durante o mestrado, especialmente o José Soares, o Ricardo Cerejeira e o André Silva bem como aos meus colegas de trabalho na Universidade Lusíada do Porto por toda a paciência dispendida para comigo.

Uma palavra de agradecimento ao meu padrinho por tudo aquilo que fez por mim e pelo apoio prestado em vários momentos da minha vida.

Dedico este trabalho igualmente aos meus pais, por tudo aquilo que ambos fizeram por mim e em especial por me terem dado a possibilidade de estudar e tirar uma licenciatura. Sem isso nunca teria chegado aqui.

Por último, uma dedicatória muito especial vai para a minha esposa, Carla, e para a minha filhota Francisca. Sei que por vezes passamos algumas dificuldades mas agradeço-vos todo o vosso amor, compreensão e ajuda. Agora teremos mais tempo os três.

A todos os outros cujo nome não foi aqui mencionado e que me apoiaram nesta longa caminhada, o meu muito obrigado.

Resumo

A nova economia, baseada essencialmente na Internet e nas novas tecnologias, coloca, cada vez mais, uma elevada pressão sobre os recursos informáticos e sobre o *staff* técnico contratado para os gerir. Numa altura em que períodos reduzidos de *downtime* de equipamentos podem levar a quebras significativas de operação e até perdas financeiras, urge encontrar soluções que procurem minimizar falhas bem como soluções que permitam uma gestão mais eficaz e eficiente dos recursos informáticos das instituições.

Sabemos já que as redes e infra-estruturas de comunicação têm um conjunto de normas e *standards* de gestão que levaram ao desenvolvimento de plataformas e sistemas de gestão poderosíssimos. No que respeita aos servidores tem vindo a ser desenvolvidas plataformas proprietárias de gestão, dada a especificidade dos mesmos, capazes de gerir quer *hardware* quer *software*. Resta uma peça do *puzzle* que tem andado desamparada mas que desempenha um papel cada vez mais fulcral no seio das instituições. Trata-se do seu próprio parque informático ou seja, os computadores pessoais, vulgarmente designados por *desktops*. Contudo, a gestão de parques informáticos tem-se revelado extremamente complexa dada a sua heterogeneidade.

Mesmo sem o impacto mediático que mais cedo ou mais tarde terá, os esforços desenvolvidos nos últimos anos para a uniformização da gestão de parques informáticos tem conhecido avanços significativos. Para tal, muito tem contribuído o DMTF, principal entidade envolvida nestes esforços e responsável pela publicação de vários *standards* nesta área.

O tema desta dissertação foca precisamente a complexidade da gestão de parques informáticos e das enormes vantagens na implementação e utilização dos *standards* de gestão actuais para tentar resolver este problema. O trabalho desenvolvido centrou-se na criação de uma plataforma de gestão centralizada, baseada em *standards*, que permite efectuar a gestão de um parque informático.

Abstract

The new economy, based essentially on the Internet and the new technologies, places, more and more, a raising pressure on the informatic resources and the staff contracted to manage them. In a time where reduced periods of downtime of equipment can lead to significant breakings of operations and even to substantial financial losses, it brings up the necessity to find solutions that will minimize fails as well as solutions that will allow a more efficient management of the informatic resources of the institutions.

We know now that data networks and communication infrastructures have a set of management standards that led to the development of powerful management systems and platforms. As far as servers are concerned, we have assisted, given their specificity, to the development of proprietary management platforms, capable to manage hardware and software. The remaining part of the puzzle, though it plays a major role in the heart of the institutions IT infrastructure, has been left aside. That part of the puzzle is its own informatic park or, as we called it, the personal computers commonly known by desktops. However, the management of desktops has shown extremely complex given its heterogeneity.

Even without the mediate impact that sooner or later it will have, the efforts developed in the last years for the standardization of such management have known significant advances. For such, a lot of work has been put by DMTF, the main entity involved in these efforts and responsible for the publication of several standards in this area.

The subject of this dissertation focuses precisely in the complexity of the IT infrastructure management and the enormous advantages in the implementation and use of management standards to solve this problem. The developed work was centered in the creation of a platform of centralized management, based on standards, that allows to effectively manage the IT infrastructure.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

1 Introdução

À medida que os parques informáticos se tornam cada vez mais heterogéneos e distribuídos, muitas organizações começam a implementar soluções de gestão centralizadas que lhes permitem gerir não só equipamentos de rede (como *switches* ou *routers*) mas também servidores de dados/aplicacionais e até *desktop PC's*. São vários os motivos para a implementação destes sistemas de gestão centralizada:

– redução dos custos operacionais

– redução dos períodos de *downtime* originados por avarias ou problemas de configuração de equipamentos.

– facilidade na gestão de diversos tipos de equipamentos

– gestão remota de *sites* geograficamente dispersos

– ausência de equipas técnicas em locais remotos

Nos últimos anos, muitas organizações adoptaram com sucesso *frameworks* de gestão como o TIVOLI da IBM [57] ou OpenView da HP [6] para gerir os seus dispositivos de rede e plataformas proprietárias para a gestão de servidores. Com o decorrer do tempo, atendendo às necessidades das empresas, estes *frameworks* e plataformas proprietárias evoluíram para soluções cada vez mais completas e abrangentes, sendo que, actualmente, é frequente disponibilizarem módulos para gestão de *storage* ou até módulos para o *deployment* de *software*, entre outros. No entanto, as organizações deparam-se hoje em dia com uma nova frente de trabalho: a gestão de equipamentos ao nível do utilizador final ou seja, o *desktop*. Cada vez mais, a dependência funcional de uma organização sobre os seus recursos informáticos estende-se para lá dos servidores e das suas redes, atingindo os *desktops*. Urge encontrar soluções que, à luz do que foi feito para as redes de comunicação, permitam gerir os actuais parques informáticos, preferencialmente de forma centralizada, tendo em conta uma das suas principais características: a heterogeneidade (quer em software quer em hardware).

As ferramentas de gestão centralizada de *desktop* são agora vistas como um requisito chave na redução do *Total Cost of Ownership* (TCO) associado ao suporte dos *desktop PC's* e como um elemento potenciador para o aumento da qualidade de serviço no utilizador final. Além disso, para alguns postos de trabalho em muitas organizações, o PC é visto como um recurso crítico que deve ser gerido como uma parte integrante do ambiente de rede em vez de apenas um recurso isolado, gerido numa base individual.

Actualmente, na maioria das organizações com alguma dimensão, é prática comum, quer devido a políticas de segurança, quer devido a políticas de funcionamento da própria organização, limitar o acesso dos utilizadores nos seus postos de trabalho à execução de determinadas aplicações e tarefas, impedindo-os de alterar ou aceder a configurações do mesmo. Toda a gestão é da responsabilidade da equipa de administração de sistemas ou administração de rede. Daqui advém a classificação de administrador que é dada à pessoa ou pessoas que fazem parte dessas equipas e que gerem a infra-estrutura. Este termo será usado no decorrer deste trabalho para identificar os técnicos responsáveis pela gestão do parque informático. Em alguns pontos, o uso do termo técnico ou técnicos tem o mesmo significado, representando apenas um grau hierárquico inferior no conjunto de pessoas afectas à equipa de informática.

Um dos grandes problemas que um administrador de redes e sistemas encontra é muitas vezes a heterogeneidade do parque informático bem como as diferentes tecnologias envolvidas e a necessidade de gerir equipamentos de vários fabricantes. No sentido de dar resposta a esta necessidade emergente, os fabricantes tem desenvolvido esforços para encontrarem interfaces, protocolos e normas que permitam levar a cabo uma gestão eficaz e independente da tecnologia sem, no entanto, descurarem aspectos como a segurança e interoperabilidade de sistemas. Como se verá adiante, pese embora o enorme esforço desenvolvido em direcção à *standardização*, não existem ainda soluções de gestão integradas de parques informáticos heterogéneos. Contudo, dada a importância crescente que os parques informáticos têm adquirido nos últimos anos, será apenas uma questão de tempo até que surjam soluções de gestão abrangentes e independentes de plataformas ou tecnologias.

1.1 Objectivos e Trabalho Desenvolvido

Os principais objectivos do trabalho aqui apresentado são, por um lado, a análise dos *standards* actuais desenvolvidos para a gestão de parques informáticos e por outro, a construção de uma plataforma de gestão que, através desses mesmos *standards*, permita a gestão de um parque informático, com todas as respectivas funcionalidades e características.

A implementação deste trabalho, permitiu avaliar por um lado a exequibilidade e por outro as vantagens na utilização destas novas interfaces, protocolos, normas e tecnologias na gestão de parques informáticos. A plataforma de gestão foi desenvolvida com base num *framework web* permitindo que a gestão possa ser efectuada a partir de qualquer local físico e de qualquer computador com acesso à Internet. As linguagens de programação bem como o *framework* e o sistema de base de dados utilizados na construção da plataforma estão disponíveis gratuitamente na Internet, permitindo assim que, no futuro, alterações ou evoluções na plataforma sejam realizadas de forma relativamente simples.

Como foi referido, embora nos últimos anos se tenha assistido a um esforço para o desenvolvimento de soluções de gestão integrada de parques informáticos heterogéneos, essas soluções não são ainda uma realidade pelo que o trabalho desenvolvido no âmbito desta tese se centrou na criação de uma plataforma de gestão baseada em *Windows Management Interface* [43] (WMI), tecnologia proprietária da Microsoft e suportada nativamente pelos seus sistemas operativos. Esta tecnologia, apesar de baseada em *standards* como o *Common Information Model* [16] CIM, disponibiliza um método proprietário para acesso aos dados de gestão o que impossibilita a integração com outros sistemas operativos. No entanto, e dada a estrutura modular que foi usada no desenvolvimento, o acesso aos dados de gestão é uma funcionalidade específica e independente pelo que, qualquer alteração na forma como é feito, não implica qualquer alteração a nível do comportamento da plataforma nem das funcionalidades disponibilizadas por esta.

Quanto ao conteúdo da plataforma, procurou-se desenvolver e disponibilizar as funcionalidades consideradas críticas num sistema de gestão deste tipo. Três dessas funcionalidades, porventura as de maior interesse, são a capacidade de monitorização/gestão *online*, a monitorização *offline* e o sistema de alertas e notificações. A monitorização/gestão *online* permite ao(s) administrador(es) do parque informático aceder em tempo real aos dados das máquinas e executar operações remotas sobre as mesmas, essencialmente de alteração de configurações e comportamentos. A configuração de rede ou a lista de processos em execução são exemplos de informação a que o(s) administrador(es) podem ter acesso em tempo real. A monitorização *offline*, baseada em tarefas automáticas de recolha de dados, permite ao administrador consultar o estado das máquinas, verificar históricos das mesmas e elaborar relatórios de acordo com os mais variados critérios. A título de exemplo, o(s) administrador(es) podem criar um relatório com o histórico de *hardware* de uma máquina com informação acerca dos componentes que a compõem ou que já fizeram parte da mesma. O sistema de alertas e notificações introduz no sistema a capacidade de gestão pró-activa, cada vez mais necessária actualmente. O sistema pode assim gerar alertas e notificações para um ou mais administradores o que lhes dá a possibilidade de, antecipadamente, detectar a ocorrência de falhas ou de situações anómalas e assim resolver problemas ou, pelo menos, minorar as suas consequências.

1.2 Estrutura da Tese

A presente dissertação encontra-se dividida em vários capítulos, cada um cobrindo uma área em particular do trabalho desenvolvido.

No capítulo 2 é feita uma introdução às arquitecturas e modelos de gestão. São abordadas três arquitecturas possíveis para a gestão de parques informáticos (centralizada,

distribuída e hierárquica) e é feita uma análise crítica, fundamentalmente por comparações, ao modo como essa gestão pode ser encarada.

No capítulo 3 são introduzidos os conceitos que serviram de base à prossecução deste trabalho: o modelo de informação comum (*Common Information Model*) CIM [16][17], a infra-estrutura *Web-Based Enterprise Management* [58] (WBEM), o *Windows Management Interface* [43] (WMI) e o *Simple Network Management Protocol* [8] (SNMP). Para o modelo de informação comum CIM procurou-se detalhar o máximo possível sem, no entanto, tornar o seu entendimento fastidioso, mas o suficiente para a consolidação dos conceitos inerentes ao modelo e para a percepção do seu papel como base para o trabalho desenvolvido. Quanto à infra-estrutura WBEM, são apresentados os *standards* que a compõem: "Representação de CIM em XML"[31] e "Operações CIM sobre HTTP"[38]. O WMI é apresentado como a implementação por parte da Microsoft da iniciativa WBEM e finalmente, o SNMP, como um protocolo de gestão de redes que pode também ser estendido para a área de gestão de sistemas. Apesar de útil em algumas áreas, como se verá posteriormente nos capítulos de implementação e resultados, a utilização do SNMP na área de gestão de sistemas é bastante limitada.

O capítulo 4 reflecte os detalhes abordados no desenvolvimento da plataforma de gestão, cobrindo os aspectos relacionados com a arquitectura, o sistema de informação, os módulos que a constituem e as funcionalidades disponibilizadas. São apresentadas também as ferramentas usadas no seu desenvolvimento, nomeadamente as linguagens de programação escolhidas, acompanhadas de pequenos exemplos da sua utilização no âmbito da gestão de máquinas, e ainda o *framework web* que serviu de base ao desenvolvimento da plataforma.

No capítulo 5 são apresentados os resultados obtidos a partir do funcionamento da plataforma de gestão num ambiente real mas não de produção. Para a grande maioria das funcionalidades da plataforma, são apresentados alguns *screenshots* que fornecem ao leitor uma visão global sobre a plataforma desenvolvida bem como a utilidade da mesma para gestão centralizada de parques informáticos.

O capítulo 6 contém as conclusões deste trabalho, a análise dos resultados obtidos, considerações gerais sobre os *standards* abordados no decorrer deste trabalho e perspectivas para o futuro no que respeita à continuação do desenvolvimento dos actuais *standards* e, acima de tudo, no que toca à implementação por parte dos fabricantes desses mesmos *standards* nos seus produtos.

As referências bibliográficas usadas quer no desenvolvimento do trabalho realizado quer na elaboração desta dissertação são apresentadas no capítulo 7.

Por último, acompanham esta dissertação quatro anexos, onde são apresentados com maior detalhe, aspectos técnicos relacionados com as tecnologias descritas e com o desenvolvimento da plataforma de gestão.

2 Arquitecturas e Modelos de Gestão

Num ambiente informático, o acto de gestão pode ser colocado em prática de várias maneiras. O que se pretende, neste capítulo, é dar uma perspectiva das arquitecturas mais comuns para a gestão de recursos informáticos e das formas como essa gestão pode ser feita. Depois de apresentadas as arquitecturas na secção **2.1**, suas vantagens e desvantagens, é feita uma análise aos modelos de gestão. A secção **2.2.1** confronta duas soluções de gestão: centralizada e distribuída. Em qualquer uma das duas há, inegavelmente, vantagens em fazê-lo de uma forma pró-activa ao invés da reactiva (secção **2.2.2**). Por fim, na última secção (**2.2.3**), é feita uma pequena exposição sobre a gestão em ambientes distribuídos.

2.1 Arquitectura de Gestão

Na gestão de um parque informático são vários os recursos envolvidos. Tradicionalmente, e à semelhança do que acontece na gestão de redes, existe um sistema gestor e um ou mais objectos geridos (figura 1).

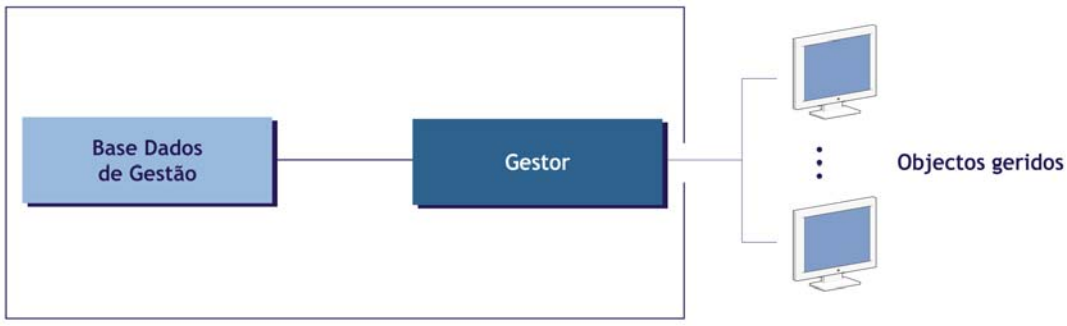


Figura 1 – Arquitectura básica de gestão

O sistema gestor corresponde ao conjunto de *software* e *hardware* que tem como função a captação e o processamento da informação de gestão. Por outro lado, o objecto gerido representa a abstracção do recurso a ser gerido. Dada a natureza heterogénea dos parques informáticos e, consequentemente, dos sistemas que os constituem, torna-se extremamente árduo tratar os recursos geridos em termos da informação de gestão. Para resolver este problema de falta de uniformidade, tem vindo a ser desenvolvidos esforços para

a criação de *standards* para a representação da informação de gestão os quais, serão vistos em maior detalhe nos capítulos posteriores.

Para que o sistema gestor e os objectos geridos possam comunicar e interagir é necessário utilizar um protocolo de comunicação que define o conjunto de acções na comunicação entre o sistema gestor e o sistema gerido. Obviamente que em casos particulares onde o sistema gestor coincide fisicamente com os recursos geridos, o protocolo de comunicação pode ser abreviado para facilitar as implementações de *software*.

2.1.1 Arquitectura de Gestão Centralizada

O modelo de gestão centralizada é uma consequência natural da utilização da topologia funcional do tipo gestor-objecto gerido. Esta arquitectura caracteriza-se pela presença de uma única estação gestora, com a total responsabilidade da gestão e uma colecção de objectos geridos. O objecto gerido interage com o sistema gestor e é representado pela sua própria informação de gestão. Esta informação de gestão, relativa a uma colecção de recursos disponibilizados pelo (ou que constituem o) objecto gerido, retrata o estado do mesmo e, através de um protocolo de comunicação, expõe-se ao sistema gestor para que este possa executar as acções de gestão necessárias. Ou seja, apesar de as funcionalidades de gestão estarem presentes nos objectos geridos, a estação gestora tem a responsabilidade da gestão dos mesmos.

A figura 2 ilustra um cenário típico de centralização da gestão.

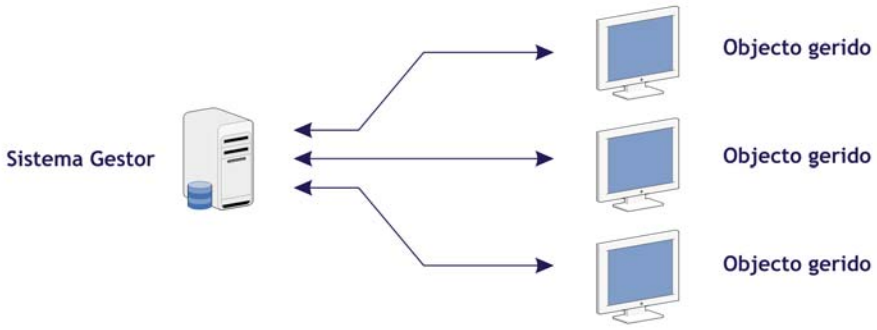


Figura 2 – Arquitectura centralizada de gestão

A vantagem deste tipo de arquitectura reside acima de tudo na sua simplicidade de implementação.

Os objectos geridos são representados por um modelo de dados que constitui uma abstracção do próprio objecto o qual, por sua vez, é gerido por um sistema central. As eventuais desvantagens deste tipo de arquitectura estão na falta de escalabilidade e na geração de tráfego de dados elevado o que pode causar congestionamentos ao nível da infra-estrutura de rede. Contudo, actualmente, estes problemas não se colocam dada a capacidade elevada no poder de processamento dos computadores actuais e as larguras de banda

consideráveis que podemos encontrar quer ao nível das redes locais quer ao nível das redes alargadas/metropolitanas. Uma outra desvantagem inerente a este tipo de solução é o facto de eventuais falhas no sistema gestor inviabilizarem por completo a gestão do parque informático.

2.1.2 Arquitectura de Gestão Distribuída

Esta arquitectura é caracterizada pela distribuição de parte das tarefas de gestão por diversos sistemas intervenientes, atribuindo-lhes desta forma um certo grau de independência. O grau e tipo de descentralização introduzido são consequência das funcionalidades que se pretendem transferir para os vários sistemas. Esta descentralização da gestão assume uma distribuição e delegação de tarefas a outras entidades. A gestão por delegação (*Management by Delegation*, ou MbD) [1] é um claro exemplo de gestão distribuída. Na figura 3 é apresentado um cenário para um ambiente com gestão distribuída.

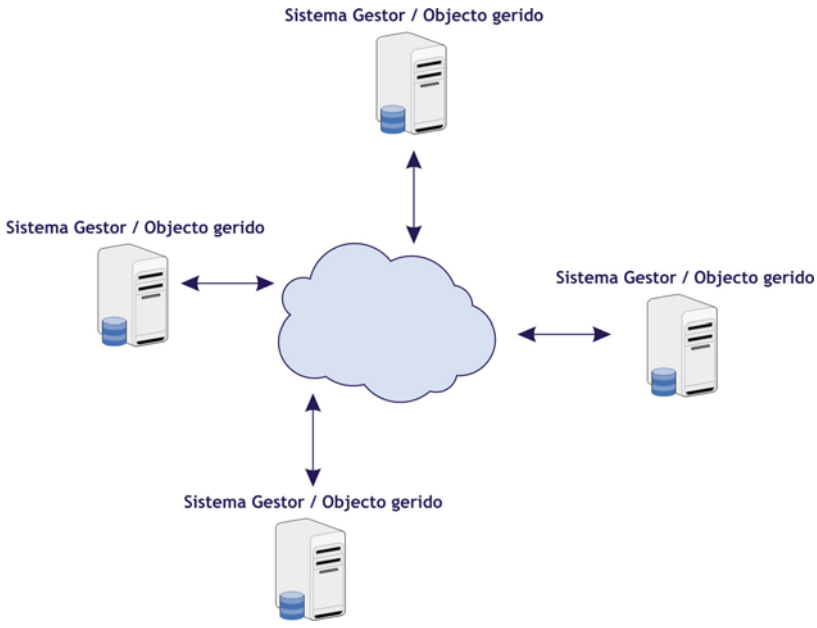


Figura 3 – Arquitectura de gestão distribuída

Face ao aumento de dimensão dos parques informáticos, que podem facilmente chegar a centenas ou milhares de postos de trabalho, uma arquitectura de gestão distribuída é mais atractiva face à cada vez maior necessidade de escalabilidade. Esta arquitectura, apesar de melhorar significativamente a escalabilidade e de ser tolerante a eventuais falhas na infra-estrutura de comunicação, apresenta a desvantagem de proporcionar aos utilizadores dos recursos um certo grau de responsabilidade o que por vezes, poderá ser inconveniente. Outra desvantagem para uma arquitectura de gestão distribuída é o aumento da complexidade nas tarefas de implementação e gestão dos sistemas de gestão (e.g. actualizações no sistema de gestão implicam actualizações em todos os nós).

2.1.3 Arquitectura de Gestão Hierárquica

Neste tipo de arquitectura, o sistema de gestão é caracterizado por uma estrutura de gestores em árvore, sendo que os sistemas geridos encontram-se no nível mais baixo. No topo, a aplicação de gestão delega tarefas nos níveis inferiores. Estes gestores de nível inferior têm a responsabilidade de executar sobre os sistemas geridos as tarefas que lhes foram delegadas pelo sistema de gestão do nível imediatamente superior. A figura 4 ilustra este tipo de arquitectura.

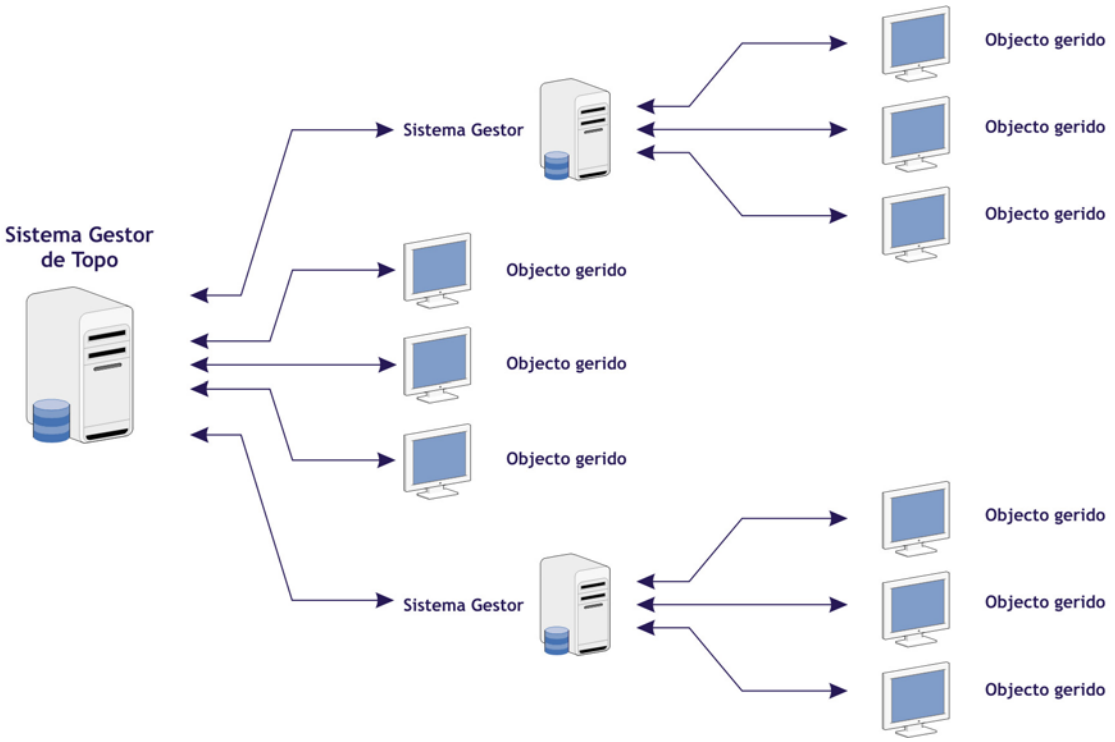


Figura 4 – Arquitectura de gestão hierárquica

Um exemplo concreto de uma arquitectura de gestão hierárquica num parque informático é o caso em que existem instalações geograficamente dispersas e em cada uma dessas instalações existe um sistema gestor cuja responsabilidade é a recolha de informação de gestão dos objectos geridos que se encontram no mesmo local geográfico. Desta forma, só existe transferência de informação de gestão entre locais físicos remotos (entre o sistema gestor de topo e o sistema gestor do local remoto) quando houver um pedido de informação por parte do sistema gestor de topo. Esta diminuição de tráfego de gestão entre entidades gestoras pode-se tornar crucial, especialmente quando a comunicação entre elas implica a ocupação do canal de comunicação entre ambos os locais.

Uma desvantagem da arquitectura de gestão hierárquica é o aumento substancial de complexidade e configuração dos sistemas gestores bem como da sua implementação e actualização.

2.2 Modelos de Gestão

Tal como existem vários tipos de arquitecturas, a gestão pode também ser encarada sob várias perspectivas. Uma gestão centralizada ou uma gestão distribuída assenta obviamente na respectiva arquitectura. No entanto, em qualquer uma das arquitecturas apresentadas, o acto de gestão poderá ser reactivo ou pró-activo. Nas secções seguintes são analisadas algumas abordagens que podem ser feitas na gestão de parques informáticos.

2.2.1 Gestão Centralizada vs Gestão Distribuída

Suponhamos que um conjunto de pessoas se encontra numa sala de reuniões e que para algumas o ambiente está demasiado quente enquanto para outras está um pouco frio. No entanto, o ajuste do termóstato de controlo de temperatura não funciona ou é ineficaz pois o controlo da temperatura é feito centralmente de acordo com determinadas regras ou condições predefinidas. De facto, este é um bom exemplo do que poderá ser a intersecção entre as regras de uma gestão distribuída e uma gestão centralizada [2].

Considere-se o caso em que o proprietário do edifício possui múltiplos aparelhos de controlo de temperatura que controlam a mesma em diferentes secções mas não em gabinetes ou espaços individuais e que estes se encontram ligados a uma unidade central de regulação e gestão. Se fosse permitido a esses gabinetes ou espaços individuais controlar a temperatura, tal resultaria em inúmeros sinais (na sua maioria distintos) enviados para a unidade central de controlo o que impossibilitaria uma harmonização no controlo da temperatura do edifício. Suponhamos então que não é permitido o ajuste da temperatura ao nível dos gabinetes ou espaços individuais e o controlo da mesma é feito centralmente por uma unidade central de gestão. Através da monitorização e controlo central da temperatura e dos requisitos das várias secções, o sistema gestor poderá optimizar os padrões de aquecimento e arrefecimento e eventualmente reduzir os custos de operação. Por exemplo, o gestor poderá configurar o sistema de arrefecimento de modo a compensar o facto de o lado Este do edifício aquecer mais rapidamente de manhã devido ao nascer do sol. Em termos de eficiência, esta é sem dúvida a melhor solução. No entanto, a solução apresenta algumas desvantagens do ponto de vista da satisfação pessoal de cada indivíduo. Suponhamos que alguém se sente desconfortável com a temperatura na sua secção. Dado que os aparelhos de aquecimento ou arrefecimento são controlados centralmente, provavelmente nunca terá as condições desejadas. O que aconteceria se se tratasse por exemplo de um membro de um conselho de administração?

Consideremos agora o cenário oposto. Cada gabinete ou secção tem o seu próprio equipamento de controlo de temperatura o qual pode ser controlado individual e localmente. Com esta configuração, não há nenhum método ou mecanismo central de controlo que

permita realizar ajustes devido a flutuações na temperatura ou eventuais compensações devidas a fenómenos externos (e.g. o nascer do sol no lado este). Neste caso, os utilizadores terão a responsabilidade de ajustar os seus sistemas por forma a satisfazer as suas necessidades o que poderá ser conveniente se estes estiverem nos seus gabinetes ou secções mas totalmente inconveniente se estiverem ausentes já que nesse caso, as condições ambientais alteram-se. Por outras palavras significa que um edifício com o controlo descentralizado da temperatura dá aos utilizadores mais flexibilidade no controlo da mesma e nas suas condições atmosféricas de trabalho mas, por outro lado, pode reduzir substancialmente a eficiência do sistema global de aquecimento/arrefecimento e, consequentemente, aumentar os custos de operação.

O ideal seria, com certeza, reunir o melhor dos dois cenários.

Transportando este exemplo para o mundo real, ou seja, para um parque informático empresarial, devemos, acima de tudo, analisar os dois cenários, considerar os benefícios de cada um deles, os problemas inerentes a cada um, cruzá-los com eventuais condicionantes de operação da empresa/instituição e, de acordo com a filosofia de operação da mesma e, em muitos casos, colocando as referidas condicionantes à frente de quaisquer outros factores, tentar conjugar os dois cenários.

De facto, aquilo que presenciamos hoje em dia, principalmente nas médias e grandes empresas é a adopção, sem qualquer dúvida do primeiro cenário apresentado ou seja, existem um conjunto de regras a cumprir que são definidas por políticas de operação tipicamente implementadas através de um sistema central de gestão. Relativamente às pequenas empresas encontramos, na grande maioria dos casos o segundo cenário, a descentralização. A principal razão desta diferença é o custo de operação e manutenção dos respectivos parques informáticos. Nas grandes e médias empresas encontramos frequentemente um *staff* técnico (ou, eventualmente, um regime de *outsourcing*) cuja responsabilidade é gerir as infra-estruturas pelo que, quanto mais centralizado for o controlo das mesmas, mais reduzida será a equipa de controlo logo, menor será o custo de operação e manutenção das referidas infra-estruturas. Por outro lado, nas pequenas empresas, muitas vezes a sua dimensão em termos de infra-estrutura e parque informático é tão reduzida que não faz sentido manter uma equipa para a gestão das mesmas forçando assim os próprios utilizadores a “controlar” os recursos com que lidam, distribuindo a responsabilidade.

2.2.2 Gestão Reactiva vs Gestão Pró-Activa

A gestão de sistemas num ambiente corporativo, envolvendo várias plataformas de *software* e vários sistemas de *hardware* pode-se tornar uma tarefa árdua e dispendiosa quer em termos de recursos quer em termos financeiros.

Para uma organização com alguma dimensão, um tempo de quebra de 30 minutos na disponibilização de um ou mais serviços por avaria ou funcionamento incorrecto de um sistema (servidor, equipamento de rede, computador pessoal, etc.) pode originar elevadas perdas competitivas e de operação pelo que se torna impraticável reagir aos problemas depois de eles acontecerem. Esta solução, designada tipicamente por gestão reactiva é, desde logo, colocada de parte em organizações com dimensão suficiente para terem um departamento ou uma equipa informática de suporte. No entanto, para a grande maioria das pequenas organizações, dada a sua reduzida dimensão, esta é normalmente a solução. Devido à carência de recursos, quer a nível de equipamentos quer a nível humano, os problemas são normalmente detectados e resolvidos após a sua ocorrência.

Todavia, nas grandes organizações, onde o preço de *downtime* é demasiado elevado, é necessário tomar as medidas necessárias para evitar quebras de serviço e no caso de estas ocorrerem, rapidamente as corrigir identificando correctamente quer a origem do problema quer a forma de o resolver [3]. É a chamada gestão pró-activa. Aqui, existe uma necessidade de acelerar os tempos necessários para a reposição do serviço, normalmente designado por *Mean-Time-To-Repair* (MTTR) bem como aumentar os períodos de operação sem falhas, designado por *Mean-Time-Between-Failures* (MTBF). Estas duas medidas, juntamente com a taxa de defeitos e o tempo médio para a ocorrência de uma falha são as medidas de confiança mais usadas [4]. A tabela 1 mostra uma definição informal dessas medidas.

Tabela 1 – Medidas de confiança

Taxa de defeitos (*failure rate*)	Número médio esperado de defeitos para um período de tempo.
MTTF (*mean time to failure*)	Tempo médio esperado até à ocorrência da primeira falha.
MTTR (*mean time to repair*)	Tempo médio esperado para reparação.
MTBF (*mean time between failures*)	Tempo médio esperado entre a ocorrência de falhas

Actualmente, o funcionamento das organizações está cada vez mais dependente das novas tecnologias e das infra-estruturas que as suportam pelo que a importância de manter um *uptime* elevado é óbvia. Contudo, o uso dessas novas tecnologias e infra-estruturas, acompanhado de uma procura incessante do aumento de competitividade, normalmente associado à diminuição dos custos de operação, acarreta outros problemas que colidem com os requisitos para o MTTR e o MTBF. Como exemplo, temos o elevado grau de dispersão geográfica presente nas organizações actuais o que, por si só, já constitui um aumento das probabilidades de ocorrer uma falha. Outro exemplo é o aumento da densidade de funcionalidades presentes num único dispositivo. Actualmente, existe a tendência para tornar os sistemas mais completos e versáteis através do aumento das funcionalidades

disponibilizadas pelos mesmos. Este aumento na densidade funcional de um dispositivo aumenta, obviamente, a probabilidade de ocorrência de falhas.

Sintetizando, tendo em conta a dependência actual das organizações nas novas tecnologias, rapidamente se evoluirá para plataformas de gestão pró-activas, não só ao nível das grandes organizações mas também nas pequenas e médias empresas. Será impensável, dentro de um curto espaço de tempo, que as organizações não tomem medidas não só para evitar falhas como apressar a correcção das mesmas. Para isso, os grandes intervenientes no mercado, sejam os fabricantes de *software* de gestão de sistemas, sejam os fabricantes de sistemas operativos ou outros, acabarão por integrar ferramentas próprias para o efeito. O trabalho desenvolvido no âmbito desta tese de mestrado pretende, essencialmente, dar a conhecer e mostrar que através da adopção de *standards* não proprietários e de normas abertas, o caminho para este tipo de gestão poderá ser abreviado.

2.2.3 Gestão de Ambientes Distribuídos

Segundo Westerinen e Bumpus [5], a evolução é um dado adquirido e os sistemas informáticos não fogem a esta regra. De facto, na década de 60, o problema da gestão distribuída era inexistente. Os sistemas eram centralizados e instalados tipicamente num único local físico da organização. A conectividade entre sistemas era praticamente nula e a informação era trocada normalmente em suporte magnético (*tapes*). A gestão deste tipo de ambiente era portanto uma actividade centralizada em todos os aspectos (físico e lógico). Contudo, ao longo do tempo, o nível e a complexidade de conectividade entre sistemas cresceu e as redes desenvolveram-se. Na década de 80 deu-se início a uma revolução em todos os sentidos com o aparecimento dos computadores pessoais. Surgiram então as primeiras redes locais, propiciando o desenvolvimento de novas aplicações e serviços distribuídos. Adquiridos inicialmente numa base departamental, com o objectivo de executar tarefas específicas, rapidamente se apercebeu que havia a necessidade de interligar os computadores pessoais com os sistemas de informação centrais para a partilha e integração de dados. Esta mudança alterou radicalmente todo o conceito de gestão praticado até à altura. A gestão dos sistemas da organização tinha-se estendido para lá da “sala de sistemas” e caminhava até ao utilizador final ou seja, até aos computadores pessoais que se encontravam distribuídos pela organização.

No final dos anos 80, o grau de complexidade na gestão destes ambientes requeria equipas alargadas e qualificadas, fazendo disparar os custos de operação. Se por um lado, os *mainframes* eram gradualmente substituídos por sistemas mais pequenos e de funcionalidades específicas (servidores, tipicamente sistemas UNIX), por outro lado, a complexidade das infra-estruturas aumentava dada a proliferação dos computadores pessoais. Tornou-se então

necessário subdividir a "gestão da informática". Partindo dos *mainframes*, surgiram três grandes áreas de gestão: servidores, rede e *desktops*.

Para a gestão dos servidores, desenvolveram-se aplicações e plataformas de gestão proprietárias que consistiam, normalmente, na instalação de *software* extra nos sistemas e cuja responsabilidade era não só a monitorização como a capacidade de actuar sobre os recursos do próprio sistema. Estas aplicações permitem aos administradores de sistemas ter um conhecimento rigoroso do estado dos sistemas bem como serem alertados em consequência de eventuais falhas. São exemplos deste tipo de plataformas o HP OpenView [6] ou o CA Unicenter [7].

Relativamente à gestão das redes, surgiu em 1990 um protocolo especialmente desenvolvido para o efeito: *Simple Network Management Protocol* (SNMP) [8]. Este protocolo tornou-se então no *standard* para a gestão das infra-estruturas de rede. Contudo, e dadas algumas limitações de segurança existentes nas primeiras versões do protocolo (versão 1 e 2 [9][10][11]), a sua utilização rapidamente se centrou numa das suas maiores capacidades, a monitorização de equipamentos de rede. No entanto, a versão 3 introduziu mecanismos de segurança [12][13], resolvendo precisamente essas limitações. Este assunto será abordado com mais detalhe na secção **3.6**.

Faltava apenas a gestão dos equipamentos na ponta ou seja, os computadores pessoais dos utilizadores finais (*desktops*). Com o alastramento destes, rapidamente se percebeu que eram necessários *standards* e sistemas de gestão específicos para este tipo de equipamentos. Apesar de o desenvolvimento de soluções de gestão a este nível se ter iniciado tardiamente, a constituição da *Distributed Management Task Force* (DMTF) [14] em meados da década de 90 marcou o ponto de viragem em termos da gestão de *desktops* com o lançamento de vários *standards* de gestão como o *Distributed Management Interface* (DMI) [15] ou o *Common Information Model* (CIM) [16][17]. O aparecimento destes *standards* e de outras iniciativas a este nível veio permitir às organizações a implementação de soluções de gestão centralizadas em ambientes distribuídos.

Exceptuando a gestão dos servidores, os outros dois tipos de gestão cedo caminharam para soluções de gestão centralizadas dadas as vantagens inerentes a este tipo de solução. Por um lado, torna possível que o *staff* técnico possa gerir, a partir de um local central, todas as infra-estruturas de rede bem como os computadores pessoais distribuídos geograficamente pelos vários locais de trabalho das instituições. Por outro lado, a centralização das operações de gestão permitiu às organizações reduzir as suas equipas técnicas, aumentando a eficiência de operação e permitindo-lhes atingir outros níveis de competitividade e produtividade.

No que se refere aos servidores, e dadas as especificidades dos vários tipos de sistemas e serviços disponibilizados, dificilmente se caminhará para soluções de gestão centralizadas. Além disso, tipicamente, os servidores encontram-se num parque próprio com condições apropriadas para os mesmos quer físicas quer lógicas (ao nível de topologias de

rede). Quer isto dizer que, dada a sua natureza, um parque de servidores já se encontra centralizado o que facilita desde logo a sua gestão. Outro factor que diferencia a gestão de um parque de servidores a um parque de *desktops* é número de sistemas a gerir. Mesmo que o número de servidores atinja algumas dezenas, e comparando com um parque de *desktops* que por vezes atingem algumas centenas de máquinas, continua a ser mais simples instalar localmente em cada servidor *software* proprietário para a gestão do mesmo.

3 Modelos de Informação

Os modelos de informação associados à gestão de parques informáticos são apresentados neste capítulo sob a forma de *standards*. Para que a gestão deste tipo de ambientes seja feita de forma uniforme, é necessário recorrer a *standards* e normas definidas para o efeito. Nesta área, o DMTF tem representado um papel central e de elevada importância nos esforços desenvolvidos nos últimos anos para o desenvolvimento e implementação de *standards* de gestão de parques informáticos. Além de uma introdução às necessidades e vantagens do uso de *standards* e ao grupo DMTF são introduzidos, neste capítulo, os *standards* com maior aplicabilidade nesta área como o CIM e o WBEM.

3.1 Standards de Gestão

No centro de toda a flexibilidade computacional e heterogeneidade dos ambientes de trabalho das organizações, os standards desempenham um papel essencial na gestão [18]. Sem o desenvolvimento e adopção de standards, a gestão da informação torna-se mais complexa, e inviabiliza-se qualquer possibilidade de integração de sistemas. Além disso, permite que os fabricantes possam colaborar entre si, disponibilizando ao utilizador final um conjunto de informação, processos e operações consistente e integrado.

Com a necessidade de rentabilização dos investimentos realizados ao nível tecnológico, os produtos baseados em *standards* oferecem às organizações uma continuidade tecnológica e operacional, independente dos fabricantes seleccionados.

O Modelo de Informação Comum – CIM, constitui actualmente o último passo em termos de *standards* para gestão de informação independente da plataforma e/ou tecnologia. O CIM foi desenvolvido pelo DMTF, actualmente a organização que lidera o desenvolvimento, a adopção e a interoperabilidade de *standards* de gestão para organizações e Internet. O grupo é constituído por vários fabricantes de peso a nível mundial desde a área das redes à área de sistemas, propiciando a colaboração entre estes no sentido de simplificar e facilitar quer as tarefas de gestão quer a utilização das novas tecnologias que vão surgindo.

Os produtos com suporte para CIM permitem uma redução de custos associados à gestão dos ambientes distribuídos e heterogéneos implementados pelas organizações. Isto é conseguido recorrendo a uma semântica estandardizada, tal como um dicionário de termos de

gestão capazes de descrever os ambientes de rede e de computação de uma organização. Esta semântica, associada a um modelo de informação orientado a objectos possibilita uma gestão completa e global, incluindo *hardware*, *software*, sistemas e serviços bem como as relações entre estes elementos.

3.2 Distributed Management Task Force - DMTF

O DMTF é uma organização tecnológica com fins não lucrativos que lidera o desenvolvimento, a adopção e a interoperabilidade de standards de gestão para ambientes Internet e empresariais. Uma das iniciativas do grupo, o modelo de informação comum CIM, constitui-se actualmente como o único *standard* para a gestão de informação de forma independente da plataforma e/ou tecnologia, propiciando a integração e redução de custos graças à interoperabilidade entre fabricantes distintos.

Fundada em 1992, a organização reuniu vários sectores e diversos intervenientes na indústria tecnológica, formando um grupo colaborativo onde todos os membros integravam os esforços de desenvolvimento. Empresas como a 3Com, Cisco Systems, Dell Computer Corp., Hewlett-Packard, IBM, Intel, Microsoft, Nec, Novell, Oracle, Sun Microsystems, Symantec são apenas alguns dos membros desta *task force*.

Os grandes objectivos do DMTF são:

1. Incentivar a adopção de standards de gestão
2. Unificar iniciativas dos fabricantes e grupos de trabalho de gestão
3. Promover a interoperabilidade entre diferentes soluções de gestão

Como exemplo dos esforços e desenvolvimentos levados a cabo pelo DMTF, a seguinte lista contempla alguns standards públicos:

- DMI (*Desktop Management Interface*) [15]
- ASF (*Alert Standard Format*) [19]
- SMBIOS (*Systems Management BIOS*) [20]
- CIM (*Common Information Model*) (secção **3.3**)
- WBEM (*Web-Based Enterprise Management*) (secção **3.4**)
- DEN (*Directory Enabled Network*) [21]

Tendo sido o primeiro *standard* desenvolvido especificamente para a gestão de máquinas (*desktops*, *notebooks* ou servidores), o DMI disponibilizava aos fabricantes um modo consistente e não proprietário para tornar os seus produtos geríveis através de um *framework* para a gestão e *tracking* de componentes das máquinas. No entanto, devido aos rápidos avanços de novas tecnologias ao nível do DMTF, o *standard* foi já descontinuado.

A especificação ASF define interfaces de alerta e controlo remoto para máquinas sem sistema operativo tornando-as comunicáveis. A especificação SMBIOS define o modo como os fabricantes de *motherboards* e sistemas devem apresentar a informação de gestão relativa aos seus produtos num formato *standard*, estendendo a interface BIOS nos sistemas Intel.

O *standard* CIM e a iniciativa WBEM serão descritas em detalhe nas secções seguintes e constituem o núcleo de desenvolvimento do DMTF ou, pelo menos, são as iniciativas de maior impacto.

Quanto ao DEN, trata-se de uma iniciativa para o mapeamento da informação CIM em uma estrutura de directório, e.g. *Lightweight Directory Access Protocol* (LDAP) [22][23] e respectiva integração com a infra-estrutura WBEM. O DMTF pretende com este *standard* integrar a informação de gestão de sistemas com a informação empresarial já presente na respectiva estrutura de directório.

3.3 Modelo de Informação Comum - CIM

Recolher os dados de gestão de um ambiente de computação de uma organização é uma parte do problema. Um outro esforço que tem de ser feito é a normalização e organização desses dados. Nos ambientes distribuídos actuais, a capacidade do sistema de gestão detectar avarias de baixo nível (e.g. mau funcionamento de ventoinhas) em um determinado equipamento (computadores, servidores, *storage*, rede, etc.) não é suficiente. Dado que todos estes componentes constituem um ambiente cooperativo, é necessário ter em atenção a(s) consequência(s) de uma avaria: que componentes afectará, conectividade entre eles, serviços que se perderão, etc. Dado que a informação atravessa várias fronteiras, é necessário que o sistema de gestão o faça também.

Esta é precisamente a base em que assenta o CIM: uniformização na organização e representação da informação acerca dos elementos de rede e de computação que constituem os ambientes informáticos actuais e as relações existentes entre eles. O modelo de informação define e organiza uma semântica consistente e comum quer para equipamentos quer para serviços.

A organização do modelo é baseada no paradigma da orientação por objectos, promovendo o uso de heranças, relações e abstracções que melhoram a qualidade e a consistência dos dados de gestão. Esta orientação permite:

– **Abstracção e classificação:** para reduzir a complexidade, os conceitos fundamentais e os de alto nível (os objectos da gestão) são definidos claramente. Estes objectos são depois agrupados em tipos (classes) de acordo com as suas características comuns (propriedades), relações (associações) e comportamentos (métodos).

– **Herança de objectos:** além dos conceitos fundamentais e de alto nível, a criação de subclasses permite adicionar mais detalhe. Herdando toda a informação (propriedades,

métodos e associações) das classes de nível superior, as subclasses permitem criar hierarquias no modelo, colocando um nível de detalhe e complexidade na camada mais apropriada. Assim, o modelo pode ser visto como uma espécie de pirâmide onde no topo temos um objecto "fundamental" sendo que no caminho até à base, o nível de detalhe e complexidade vão aumentando progressivamente.

– **Dependências e Associações:** as relações entre objectos são conceitos extremamente poderosos. A orientação por objectos do modelo permite que as relações sejam directamente modeladas e caracterizadas por uma semântica própria sendo posteriormente designadas como associações.

– **Estandardização de Métodos:** a capacidade de definir comportamentos (métodos) universais é outra forma de abstracção. Por exemplo, veja-se a vantagem de se poder efectuar um *Reset* ou um *Reboot* remotos num qualquer equipamento independentemente do *hardware*, sistema operativo ou fabricante.

– **Reutilização de dados:** esta é uma das grandes vantagens do modelo pois permite a entrega consistente de dados de gestão independente dos produtos ou das várias versões dos mesmos. Por exemplo, um chassis é identificado pela mesma classe de objectos, quer se trate de um computador pessoal ou de um *router*.

Um outro objectivo e grande vantagem do CIM é a sua flexibilidade e suporte de novas extensões. Isto significa que um fabricante pode desenvolver o modelo de informação de modo a cobrir uma área particular. Este desenvolvimento assenta na criação de subclasses com um nível de detalhe superior. As extensões ao modelo serão vistas em maior detalhe na secção **3.3.2.3**. Um exemplo são as classes `WIN32` desenvolvidas pela Microsoft para cobrir a gestão de uma área particular: os seus sistemas operativos.

CIM é composto por dois grandes blocos: a especificação (CIM *Specification*) e o esquema (CIM *Schema*). A especificação define os detalhes para a integração com outros modelos de gestão enquanto que o esquema contém a descrição e organização do modelo. O objectivo do esquema é o de capturar noções comuns aplicáveis às várias áreas de gestão, independentemente das implementações. Nas secções que se seguem será feita uma pequena descrição de cada um destes blocos. Será ainda descrita a forma como é feita a gestão do espaço de nomes (*namespace*) para a identificação única e inequívoca de classes e instâncias de classes nos esquemas CIM.

3.3.1 CIM Specification

A especificação CIM, descrita em [16], descreve um modelo baseado no paradigma de orientação por objectos [24] e em *Unified Modeling Language* (UML) [25]. O modelo inclui classes de objectos e respectivas propriedades, métodos aplicáveis e associações.

A especificação define:

– regras e sintaxe do modelo, baseado em *Interface Definition Language* (IDL) [26] e designada por *Management Object Format* (MOF) [16, p.29];

– um mecanismo de atribuição de nomes – CIM *Naming*.

O modelo é constituído por **esquemas**, **classes**, **propriedades**, **métodos** e **qualificadores**. O modelo suporta ainda **Associações** e **Indicações** como tipos de classes e **Referências** como um tipo de propriedades.

Os **esquemas** são usados para administração e para *class naming* sendo nada mais que um grupo de classes. Cada uma dessas classes inclui o nome do *schema* no seguinte formato: *schemaname_classname*.

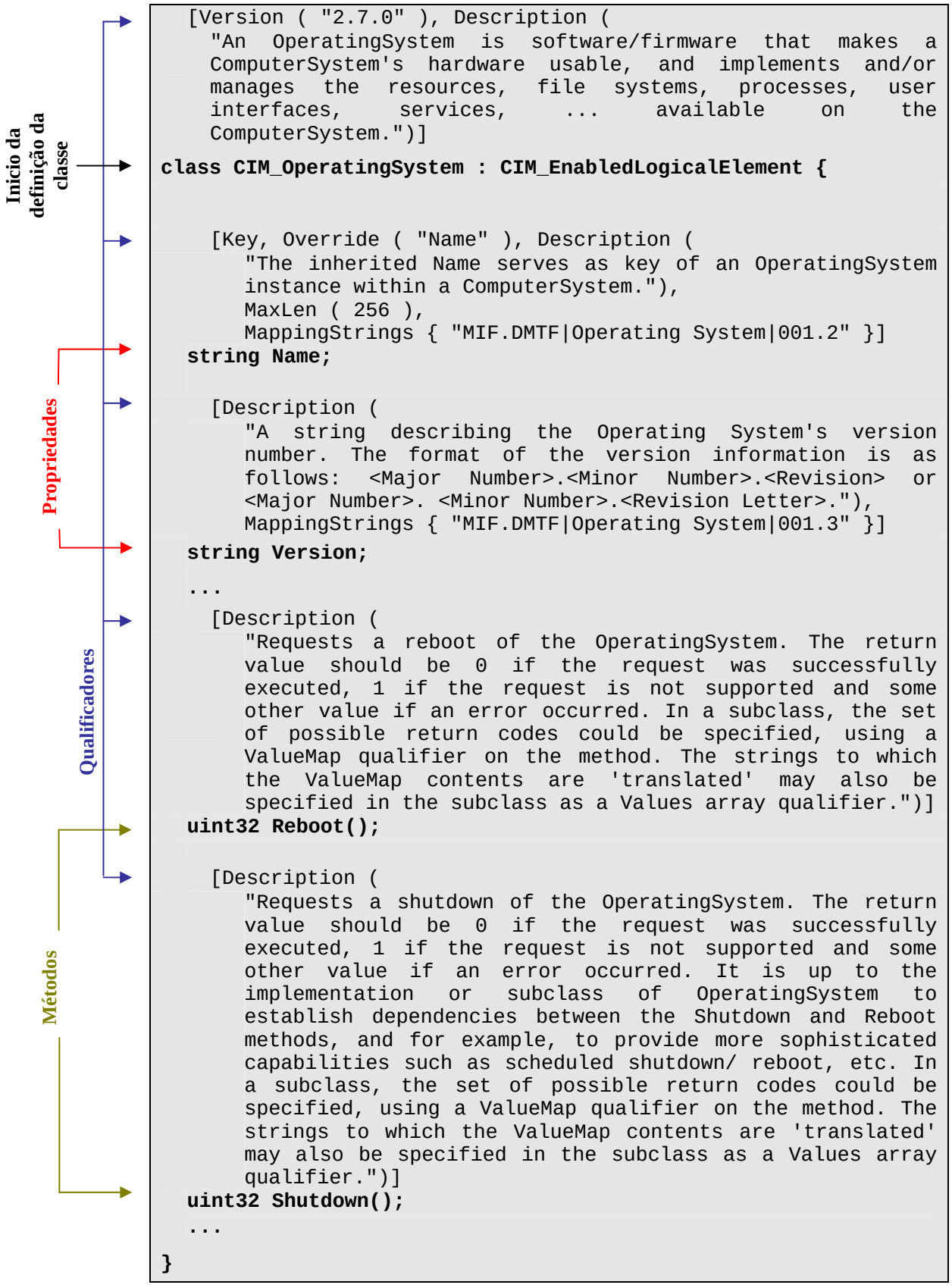
Uma **classe** é uma colecção de instâncias que suportam o mesmo tipo ou seja, as mesmas propriedades e métodos. As classes são organizadas de forma hierárquica, permitindo representar subtipos através de relações entre classes. Trata-se portanto de um protótipo que define um conjunto de propriedades e métodos comuns a um tipo de objecto. Cada classe CIM contém propriedades que representam o seu estado e métodos que descrevem o comportamento. Uma classe deve pertencer apenas a um **esquema** e o seu nome deve ser único para esse **esquema**.

Uma **propriedade** é um valor usado para denotar uma característica da classe. As propriedades têm significado local pelo que devem ser únicas apenas no interior da classe. É caracterizada por um nome, um tipo de dados e um valor. Eventualmente, poderá ter um valor por omissão. Os tipos de dados suportados encontram-se descritos em [16, p.10].

Um **método** é uma operação que pode ser invocada sobre uma instância de uma classe. Tal como as propriedades, os métodos devem apenas ser únicos dentro da classe. Um método retorna normalmente um valor e pode aceitar vários parâmetros. Quer o valor de retorno quer os parâmetros devem ter um tipo de dados suportado pela norma (tal como as propriedades).

Os **qualificadores** são valores usados para fornecer informação adicional acerca das classes, associações, indicações, métodos, propriedades ou referências. O qualificador só pode ser usado a partir do momento em que é definido o tipo de qualificador (*qualifier type definition*). É caracterizado por um nome, um tipo, um valor, um *scope*, um *flavor* e opcionalmente um valor por omissão. O tipo do qualificador pode ser qualquer um dos definidos para as propriedades. O *flavor* define comportamentos adicionais para os qualificadores. Por exemplo, um qualificador pode ser transmitido automaticamente para classes derivadas da original ou então manter-se restrito na classe original onde foi definido. O *scope* define os elementos (classe, associação, indicação, propriedade, referência, método, parâmetro) aos quais o qualificador é aplicado. Pelo menos um elemento é obrigatório podendo, no entanto, conter vários elementos.

Consideremos o seguinte exemplo para a classe `CIM_OperatingSystem`, pertencente ao esquema CIM, onde, para simplificação, são apresentadas duas das suas propriedades (`Name` e `Version`) e dois dos seus métodos (`Reboot()` e `Shutdown()`).



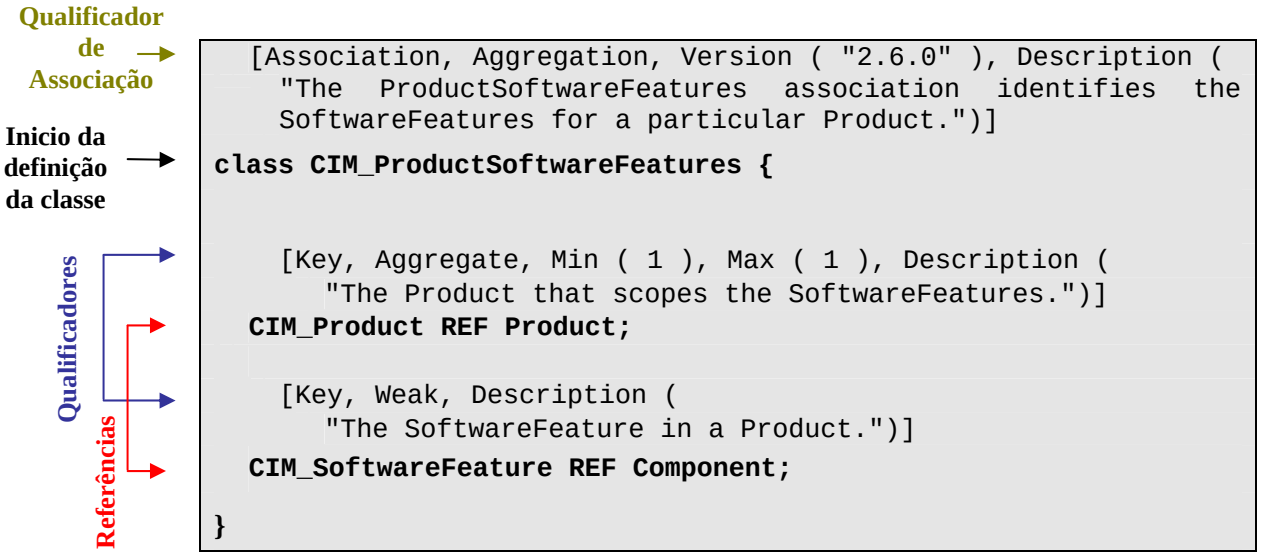

```
   [Version ( "2.7.0" ), Description (
     "An OperatingSystem is software/firmware that makes a
     ComputerSystem's hardware usable, and implements and/or
     manages the resources, file systems, processes, user
     interfaces, services, ... available on the
     ComputerSystem.")]
class CIM_OperatingSystem : CIM_EnabledLogicalElement {


     [Key, Override ( "Name" ), Description (
        "The inherited Name serves as key of an OperatingSystem
        instance within a ComputerSystem."),
        MaxLen ( 256 ),
        MappingStrings { "MIF.DMTF|Operating System|001.2" }]
   string Name;

     [Description (
        "A string describing the Operating System's version
        number. The format of the version information is as
        follows: <Major Number>.<Minor Number>.<Revision> or
        <Major Number>. <Minor Number>.<Revision Letter>."),
        MappingStrings { "MIF.DMTF|Operating System|001.3" }]
   string Version;

   ...
     [Description (
        "Requests a reboot of the OperatingSystem. The return
        value should be 0 if the request was successfully
        executed, 1 if the request is not supported and some
        other value if an error occurred. In a subclass, the set
        of possible return codes could be specified, using a
        ValueMap qualifier on the method. The strings to which
        the ValueMap contents are 'translated' may also be
        specified in the subclass as a Values array qualifier.")]
   uint32 Reboot();

     [Description (
        "Requests a shutdown of the OperatingSystem. The return
        value should be 0 if the request was successfully
        executed, 1 if the request is not supported and some
        other value if an error occurred. It is up to the
        implementation or subclass of OperatingSystem to
        establish dependencies between the Shutdown and Reboot
        methods, and for example, to provide more sophisticated
        capabilities such as scheduled shutdown/ reboot, etc. In
        a subclass, the set of possible return codes could be
        specified, using a ValueMap qualifier on the method. The
        strings to which the ValueMap contents are 'translated'
        may also be specified in the subclass as a Values array
        qualifier.")]
   uint32 Shutdown();
   ...
}
```

No exemplo apresentado, tanto a classe como as propriedades e os métodos são melhor descritas através do uso de qualificadores, imediatamente antes de cada elemento. Uma outra nota de particular interesse no exemplo apresentado é o facto de a classe `CIM_OperatingSystem` derivar da classe `CIM_EnabledLogicalElement` ou seja, tal como referido já anteriormente, o modelo CIM permite uma organização hierárquica o que permite aumentar o nível de detalhe à medida que se desce na hierarquia.

Além dos elementos apresentados (qualificadores, classe, propriedades e métodos), existem ainda as **Referências** e as **Associações**.

Uma **referência** é um tipo de propriedade especial, declarada por `REF`, indicando que se trata de um apontador para outras instâncias. Uma referência define o papel que cada objecto desempenha numa associação. Uma **associação** é um tipo especial de classe que contem duas ou mais referências e representam as relações entre duas ou mais classes. As associações são classes identificadas por um qualificador de associação. Dado que uma associação é uma classe, a adição ou remoção de uma determinada associação não tem qualquer efeito sobre as classes relacionadas.

O exemplo seguinte representa uma classe de associação, caracterizada apenas por duas propriedades, neste caso, referências. Trata-se da classe `CIM_ProductSoftwareFeatures` que associa funcionalidades de *software* com o respectivo produto.




```
[Association, Aggregation, Version ( "2.6.0" ), Description (
  "The ProductSoftwareFeatures association identifies the
  SoftwareFeatures for a particular Product.")]
class CIM_ProductSoftwareFeatures {


   [Key, Aggregate, Min ( 1 ), Max ( 1 ), Description (
      "The Product that scopes the SoftwareFeatures.")]
  CIM_Product REF Product;

   [Key, Weak, Description (
      "The SoftwareFeature in a Product.")]
  CIM_SoftwareFeature REF Component;

}
```

Segundo o modelo de informação comum CIM, um produto de *software* (`Product`) é uma colecção de funcionalidades de *software* (*software features*) que podem ser adquiridas como uma unidade. Por seu lado, uma funcionalidade de *software* (`SoftwareFeature`) é uma colecção de elementos de *software* que executam uma função ou desempenham um papel num produto de *software*. Neste sentido, a classe apresentada no exemplo, `CIM_ProductSoftwareFeatures`, representa a associação entre funcionalidades de *software* e o respectivo produto de *software*.

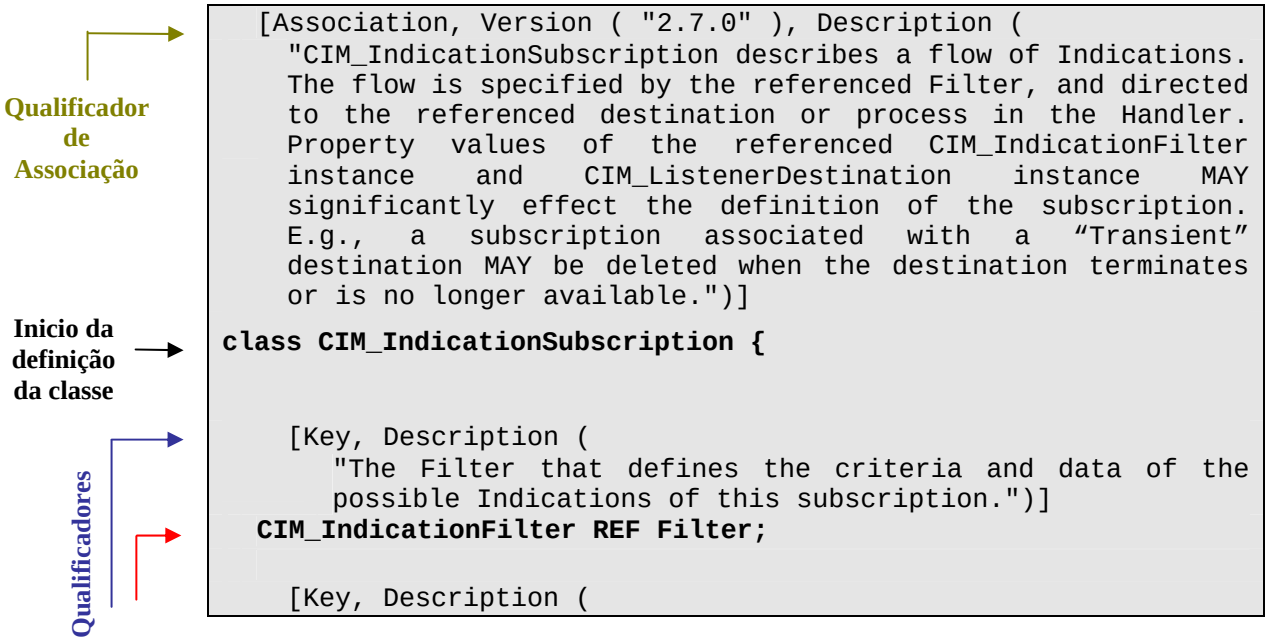
3.3.1.1 Tratamento de Eventos

Dada a especificidade do tratamento de eventos em CIM e do modo como é encarado, nesta subsecção será feita uma descrição do modelo de eventos em maior detalhe. O modelo de eventos CIM define abstracções relacionadas com os eventos: uma hierarquia de **Indicações** e o modo como estas são usadas para modelar **Eventos**. O modelo define ainda o uso de **Subscrições** como forma de registo para a recepção de **Indicações**.

Segundo o modelo CIM, um evento pode ser definido como uma mudança de estado (e.g. alteração de estado de um serviço) ou, alternativamente, como a ocorrência de um fenómeno de interesse (ex.: erro de escrita em disco). Apesar de os eventos serem modelados através de indicações existe uma diferença fundamental entre ambos. Um evento pode ser definido como a ocorrência de um fenómeno de interesse enquanto uma indicação é o registo da detecção de um evento de interesse. Não existe, portanto, uma correspondência um-para-um entre eventos e indicações. Por exemplo, um fenómeno de interesse poderia ser a falha de uma memória. Poderiam também ser geradas várias indicações, relativas à detecção de múltiplos erros devido a falhas em subcomponentes da memória. No entanto, cada um desses erros estaria associado ao mesmo evento: falha do componente memória.

A ocorrência de um evento é representada por uma instância da classe `CIM_Indication`. O modelo assume que as indicações apenas podem ser recebidas por clientes se estes as subscreverem ou seja, se efectuarem um registo prévio. Uma **Subscrição** é expressa pela criação de uma instância de uma associação do tipo *IndiccationSubscription* que referencia um filtro (*IndicationFilter*) e um *handler* (*ListenerDestination*). O filtro contém a *query* que seleccionará os dados enquanto que o *handler* identifica o consumidor do evento.

A representação em sintaxe MOF de uma subscrição é apresentada de seguida.

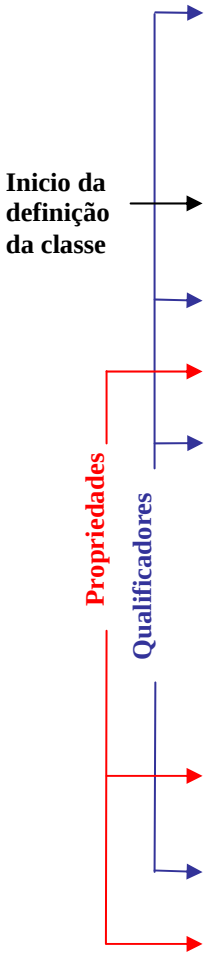



```
[Association, Version ( "2.7.0" ), Description (
  "CIM_IndicationSubscription describes a flow of Indications.
  The flow is specified by the referenced Filter, and directed
  to the referenced destination or process in the Handler.
  Property values of the referenced CIM_IndicationFilter
  instance and CIM_ListenerDestination instance MAY
  significantly effect the definition of the subscription.
  E.g., a subscription associated with a “Transient”
  destination MAY be deleted when the destination terminates
  or is no longer available.")]
class CIM_IndicationSubscription {


  [Key, Description (
    "The Filter that defines the criteria and data of the
    possible Indications of this subscription.")]
CIM_IndicationFilter REF Filter;

  [Key, Description (
```


```
        "The Handler addressing delivery of the possible
        Indications of this subscription.")]
    CIM_ListenerDestination REF Handler;

    ...

}
```

A classe `CIM_IndicationSubsccription` é caracterizada por um conjunto de propriedades sendo apenas apresentadas aquelas de especial interesse, neste caso, as referências para o filtro, `CIM_IndicationFilter`, e para o *handler*, `CIM_ListenerDestination`.

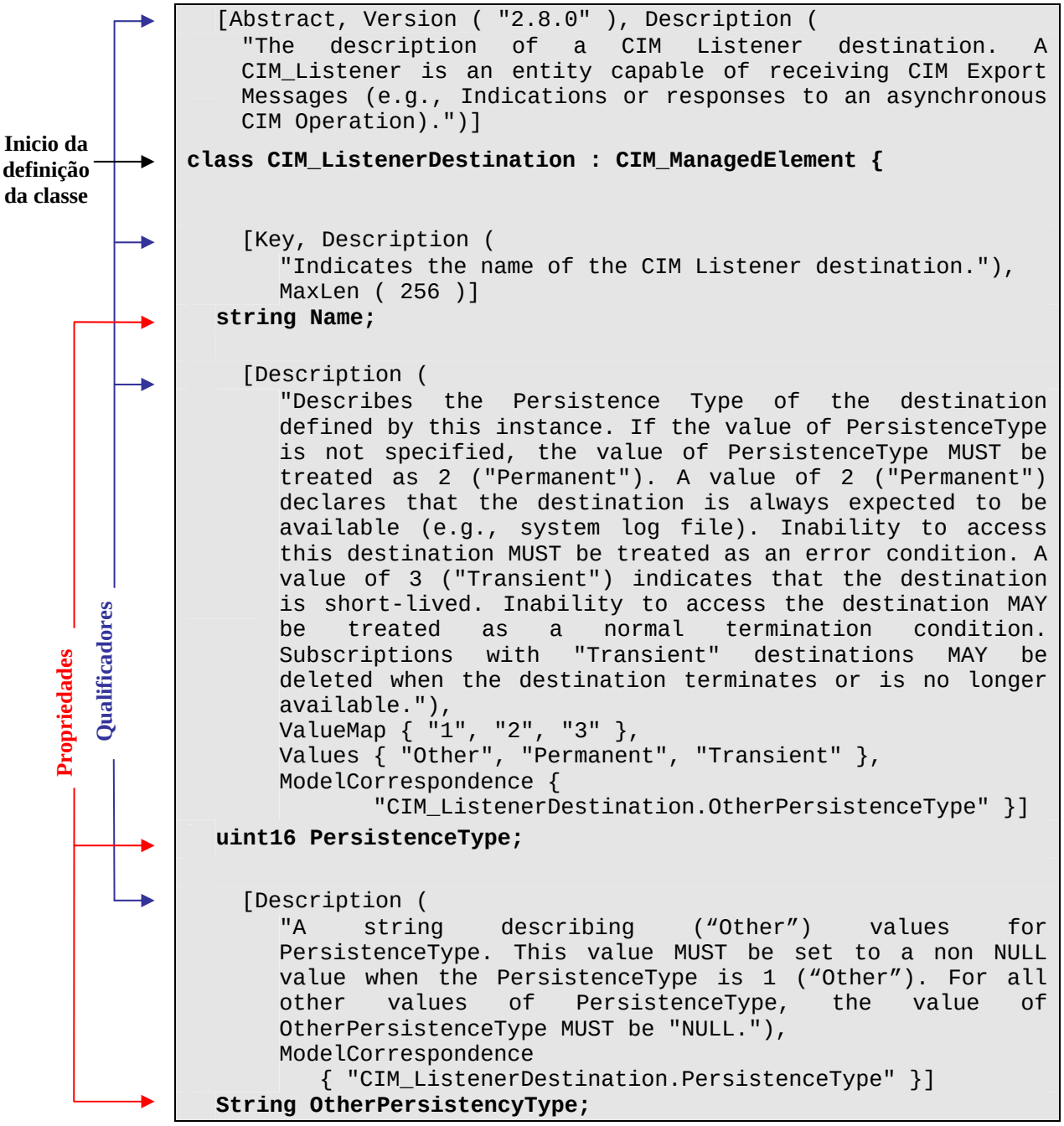
A descrição do filtro é feita pela classe `CIM_IndicationFilter`.




```
  [Version ( "2.6.0" ), Description (
    "CIM_IndicationFilter defines the criteria for generating an
    Indication and what data should be returned in the
    Indication. It is derived from CIM_ManagedElement to allow
    modeling the dependency of the filter on a specific
    service.")]
class CIM_IndicationFilter : CIM_ManagedElement {


    [Key, Description (
       "The name of the filter.")]
  string Name;


    [Required, Description (
       "A query expression that defines the condition(s) under
       which Indications will be generated. For some Indication
       classes, the query expression may also define the
       instance properties to be copied to the
       CIM_InstIndication's SourceInstance and PreviousInstance
       properties. Query language semantics include projection
       (e.g., Select), range (e.g., From) and predicate (e.g.,
       Where)."),
       ModelCorrespondence
       { "CIM_IndicationFilter.QueryLanguage" }]
  string Query;


    [Required, Description (
       "The language in which the query is expressed.")]
  string QueryLanguage;

  ...

}
```

Entre outras coisas, um filtro é caracterizado por um nome, uma *query* e uma *query language*. Como já foi dito, a *query* define as condições sob as quais será gerada uma indicação. A propriedade *query language* indica a linguagem usada para expressar a *query*.

A identificação do destinatário da subscrição é feita através da classe `CIM_ListenerDestination`.




```
   [Abstract, Version ( "2.8.0" ), Description (
     "The description of a CIM Listener destination. A
     CIM_Listener is an entity capable of receiving CIM Export
     Messages (e.g., Indications or responses to an asynchronous
     CIM Operation).")]
class CIM_ListenerDestination : CIM_ManagedElement {


     [Key, Description (
        "Indicates the name of the CIM Listener destination."),
        MaxLen ( 256 )]
   string Name;

     [Description (
        "Describes the Persistence Type of the destination
        defined by this instance. If the value of PersistenceType
        is not specified, the value of PersistenceType MUST be
        treated as 2 ("Permanent"). A value of 2 ("Permanent")
        declares that the destination is always expected to be
        available (e.g., system log file). Inability to access
        this destination MUST be treated as an error condition. A
        value of 3 ("Transient") indicates that the destination
        is short-lived. Inability to access the destination MAY
        be treated as a normal termination condition.
        Subscriptions with "Transient" destinations MAY be
        deleted when the destination terminates or is no longer
        available."),
        ValueMap { "1", "2", "3" },
        Values { "Other", "Permanent", "Transient" },
        ModelCorrespondence {
              "CIM_ListenerDestination.OtherPersistenceType" }]
   uint16 PersistenceType;


     [Description (
        "A    string    describing    ("Other")    values    for
        PersistenceType. This value MUST be set to a non NULL
        value when the PersistenceType is 1 ("Other"). For all
        other   values   of   PersistenceType,   the   value   of
        OtherPersistenceType MUST be "NULL."),
        ModelCorrespondence
           { "CIM_ListenerDestination.PersistenceType" }]
   String OtherPersistencyType;
```

Entre outras características, o destinatário de uma subscrição tem um nome (`Name`) e um tipo de persistência (`PersistenceType`) que, de acordo com a explicação apresentada no respectivo qualificador, indica se o destinatário tem um carácter permanente (e.g. ficheiro de *log* de sistema) ou transitório.

3.3.2 CIM Schema

O modelo CIM faz com que o ambiente gerido possa ser visto como uma colecção de sistemas interrelacionados, cada um dos quais composto por um número discreto de elementos. O esquema CIM está organizado numa série de classes com determinadas

propriedades e associações que se constituem como um *framework* no qual é possível organizar toda a informação disponível sobre o ambiente gerido.

O esquema CIM encontra-se dividido em três blocos:

– ***Modelo Nuclear ou Núcleo***

Contém noções aplicáveis a todas as áreas de gestão. Trata-se de um conjunto de classes, associações e propriedades que representam um vocabulário básico para descrever os sistemas geridos.

– ***Modelos Comuns***

São modelos de informação que contém noções comuns a determinadas áreas de gestão e independentes do tipo de tecnologia, fabricante ou implementação. As classes, propriedades, associações e métodos dos modelos comuns permitem uma vista mais detalhada, suficiente para servir como ponto de partida para o desenvolvimento de aplicações de gestão

– ***Extensões do Modelo***

As extensões representam um conjunto de informação específica de uma tecnologia, implementação ou fabricante. Por exemplo, um sistema operativo pode ter o seu conjunto de extensões para melhor o caracterizar e/ou gerir.

O objectivo é que os modelos comuns evoluam no sentido de promover e englobar objectos e propriedades definidas nas extensões, como veremos mais adiante.

3.3.2.1 Núcleo

No núcleo do modelo CIM (***Core***) é estabelecida a classificação básica dos elementos e associações do ambiente gerido. `CIM_ManagedElement` é a classe raiz da hierarquia de objectos CIM e serve como base para todas as associações que se aplicam às entidades na hierarquia. A partir das classes do núcleo, o modelo expande-se em várias direcções, abrangendo vários domínios de gestão bem como as relações entre as entidades geridas.

Os objectos definidos no núcleo definem-se e associam-se conforme a

figura 5 na página seguinte. Note-se que, para simplificar, nem todas as classes do núcleo estão incluídas na figura.

Na segunda camada da hierarquia encontramos subclasses da classe `CIM_ManagedElement` como:

– `CIM_Product`: representa contratos entre o consumidor e vendedores, contendo informação sobre a forma como o produto foi adquirido, como é suportado e onde está instalado

– `CIM_Setting`: define parâmetros específicos e pré-configurados e que são aplicados a um ou mais elementos do sistema (*Managed System Elements*)

– `CIM_Configuration`: inclui definições e dependências, representando um comportamento ou um estado funcional do(s) elemento(s) de um sistema.

– `CIM_StatisticalInformation`: é uma superclasse abstracta a partir da qual deriva qualquer classe com dados estatísticos sobre um elemento gerido (*Managed Element*). O elemento ao qual se aplica a informação estatística é indicado através da associação `CIM_Statistics`.

– `CIM_Collection`: representam grupos de elementos com características comuns

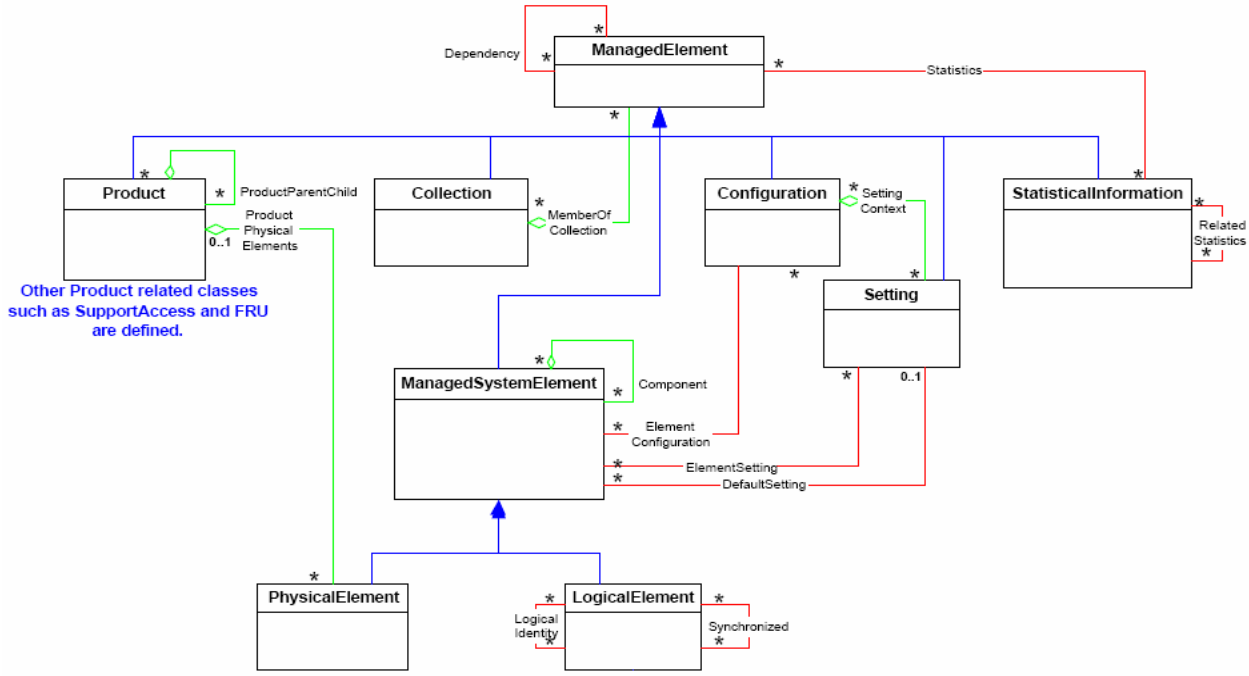


Figura 5 – Topo da hierarquia de objectos do modelo CIM (fonte [17])

Voltando à segunda camada, imediatamente abaixo da raiz, encontramos, além das referidas anteriormente, a classe `CIM_ManagedSystemElement`. As instâncias desta classe podem representar sistemas, componentes de sistemas, serviços, *software* ou redes. A definição de **Sistema** no contexto do modelo CIM é algo abrangente podendo-se referir a computadores, dispositivos dedicados, aplicações ou até domínios de rede.

As classes `CIM_PhysicalElement` e `CIM_LogicalElement` são ambas subclasses da classe `CIM_ManagedSystemElement`. A primeira representa entidades físicas de um **Sistema** enquanto a segunda representa abstracções que permitem gerir, configurar e coordenar o **Sistema** ou o *software*. Tipicamente, representam os próprios **Sistemas**, componentes do **Sistema** ou funcionalidades de *software*.

A partir destas classes, o modelo expande-se para as várias áreas de gestão.

3.3.2.2 Modelos Comuns

Tratam-se de modelos de informação que capturam características particulares de várias áreas de gestão, independentemente das tecnologias e/ou implementações. Actualmente, a versão CIM 2.7 contempla os seguintes modelos:

- Modelo de Aplicação
- Modelo de Eventos
- Modelo de Rede
- Modelo de Suporte
- Modelo de Base de Dados
- Modelo de Interoperabilidade
- Modelo Físico
- Modelo de Sistemas
- Modelo de Dispositivos
- Modelo de Métricas
- Modelo de Políticas
- Modelo de Utilizador

Vejamos, então, os objectivos de cada um destes modelos.

O **Modelo de Aplicação** contém a informação comum e necessária para o desenvolvimento e gestão de produtos de *software*. É baseado na necessidade de gerir o tempo de vida e a execução das aplicações e assenta em três conceitos essenciais: estrutura da aplicação, *lifecycle* da aplicação e a transição entre estados durante o *lifecycle* (*deployment*, instalação e configuração, *startup* e operação). Em termos de modelo CIM, uma aplicação é constituída pelos seguintes componentes:

- ***Software Product***: colecção de funcionalidades de *software* que podem ser adquiridas como uma unidade.
- ***Software Feature***: é uma colecção de elementos de *software* que executam uma determinada função ou papel no produto de *software*.
- ***Software Element***: colecção de um ou mais ficheiros e respectivos detalhes que são instalados e geridos numa base individual numa plataforma.
- ***Application System***: é uma colecção de *software features* que são geridas como uma unidade independente e que suporta uma ou mais funções.

Segundo o **Modelo de Eventos**, um evento é assumido como sendo, tipicamente, uma mudança de estado ou o registo de um comportamento de um componente de um ambiente. Por exemplo, o estado de um serviço pode-se alterar de *Stopped* para *Started*, ou um

dispositivo pode ser adicionado a uma máquina resultando numa indicação de *plug-and-play* para o sistema operativo. O modo como os eventos são tratados difere de evento para evento. Enquanto alguns requerem uma acção imediata, outros podem arrastar-se no tempo.

No contexto da especificação CIM, a ocorrência de um evento é representada por uma instância da classe `CIM_Indication` a qual se constitui como a superclasse da hierarquia de indicações. Como classe abstracta, `CIM_Indication` serve como base para todas as subclasses de indicações conforme a figura 6.

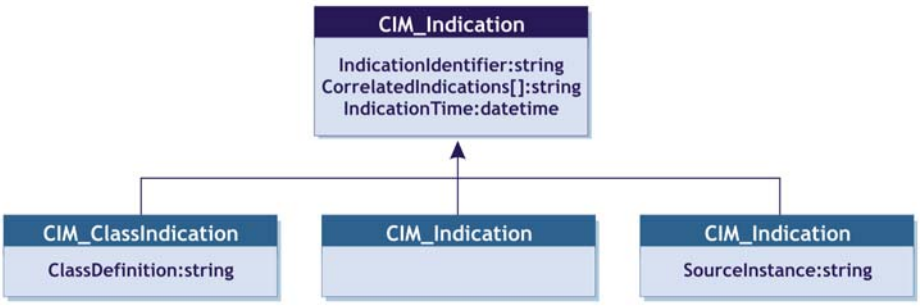


Figura 6 – Topo da hierarquia de indicações do modelo de eventos CIM

Na actual versão do esquema de eventos, existem três subclasses de `CIM_Indication`, representando três tipos de eventos:

– `CIM_InstIndication` é usado para modelar eventos CIM do género *lifecycle* (e.g. criação, alteração, remoção de instâncias ...)

– `CIM_ClassIndication` é usado para modelar eventos CIM *Schema* (e.g. criação, alteração, remoção de classes ...)

– `CIM_ProcessIndication` é usado para notificações de alertas associados a objectos que podem ou não ser completamente modelados pelos objectos CIM (e.g. alertas de baixo nível como DMI *alerts* [15] ou SNMP *traps* [27]).

Um conceito fundamental no modelo de eventos CIM é a separação entre publicações de indicações e subscrição de indicações. A publicação de uma indicação é feita através do mesmo mecanismo utilizado para publicar outro tipo de dados em CIM ou sejam, pela declaração de classes e respectivas propriedades da indicação. A publicação de eventos implica também a criação de instâncias `CIM_IndicationFilter`. Uma subscrição é expressa pela criação de uma instância da classe `CIM_IndicationSubscription` com uma referência para a instância `CIM_IndicationFilter` e outra para a instância `CIM_ListenerDestination` (ver secção **3.3.1**).

O **Modelo de Rede** é o que descreve e gere a conectividade e comunicações de uma rede, bem como serviços e protocolos disponibilizados ou transportados pela mesma. As entidades geridas pelo modelo podem ser agrupadas nas seguintes categorias:

– Dispositivos/Equipamentos de rede: inventário, informação sobre utilizadores e segurança, etc.
– Serviços de rede: roteamento, comutação, etc.
– Acesso lógico e interconectividade: *protocol endpoints*, rotas, etc.
– Protocolos de rede: OSPF [28], BGP [29], etc.
– Tecnologias de rede: *switching/bridging*, VLAN's, etc.
– Qualidade de serviço: tecnologias, filas de espera, etc.
– Outros: por exemplo critérios de filtragem de pacotes, etc.

O modelo CIM caracteriza uma rede como um tipo de domínio administrativo que pode conter, dentro de si, outras redes, sub-redes ou domínios. Para a correcta operação da infra-estrutura de rede, o modelo define também os serviços de rede. Dado o significado algo lato do termo "serviço", em CIM, um serviço refere-se a uma funcionalidade disponibilizada pela infra-estrutura de rede ou requerida pelos elementos da rede para operarem e trocarem informação entre si. Exemplos de serviços são o roteamento, encaminhamento, qualidade de serviço, etc. Dentro dos domínios administrativos (redes), existem os chamados dispositivos de rede: comutadores, roteadores, repetidores, etc. Apesar de desempenharem funções completamente distintas, os dispositivos de rede são vistos como **Sistemas**. Quer isto dizer que um roteador é visto, segundo o modelo CIM, da mesma forma que um computador ou seja, ambos são representados por instâncias da classe `CIM_ComputerSystem`.. Além de cobrir as áreas de gestão de configuração e de estado, o modelo define também o modo como poderão ser recolhidas estatísticas a partir dos elementos da rede tendo como objectivo a gestão da performance.

O **Modelo de Suporte** pretende disponibilizar aos fabricantes uma plataforma de trabalho comum que lhes permita trocar informação entre si no sentido de optimizarem soluções. Um dos princípios fundamentais para o projecto e desenvolvimento de sistemas é a sua arquitectura modular que, com aproximações do tipo "*plug-and-play*", permite que diversos produtos de vários fabricantes operem em conjunto.

Os utilizadores finais sentem cada vez mais a necessidade de usar produtos de vários fabricantes e esperam, acima de tudo, que eles sejam interoperáveis. Contudo, sem uma forma estandardizada de representar e disponibilizar a informação de cada um dos fabricantes, o processo de recolha, publicação e interpretação dessa mesma informação torna-se numa tarefa extremamente árdua, inconsistente e completamente ineficaz. Este modelo pretende servir como base para essa troca de informação.

O **Modelo de Base de Dados** define os componentes de gestão para um sistema de base de dados. Conceptualmente, o modelo baseia-se em três grandes entidades:

– ***Database System***: representa as características aplicacionais do software de base de dados, e.g. organização, armazenamento, segurança, gestão, etc.

– ***Common Database***: entidade lógica que representa os dados e a sua organização

– ***Database Service***: representa o processo ou processos que coordenam o acesso dos utilizadores à base de dados, e.g. autenticação, autorização, concorrência, manipulação de dados, verificação de integridade, etc.).

A figura 7 representa conceptualmente um sistema de base de dados.

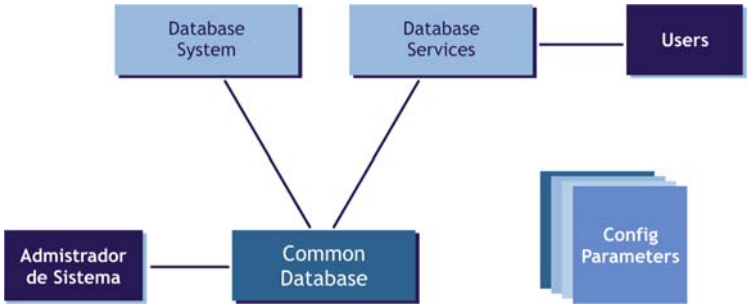


Figura 7 – Sistema de base de dados segundo o modelo CIM

O **Modelo de Interoperabilidade** define os componentes de gestão que descrevem a infra-estrutura *Web Based Enterprise Management* (WBEM) e o modo como os seus componentes (e.g. *providers* e *protocol adapters*) interagem com a infra-estrutura. O WBEM é um *standard* desenvolvido pelo DMTF e será descrito em maior detalhe na secção seguinte. A infra-estrutura WBEM é composta essencialmente por três blocos (ver figura 8):

– cliente CIM

– servidor CIM

– elementos geridos

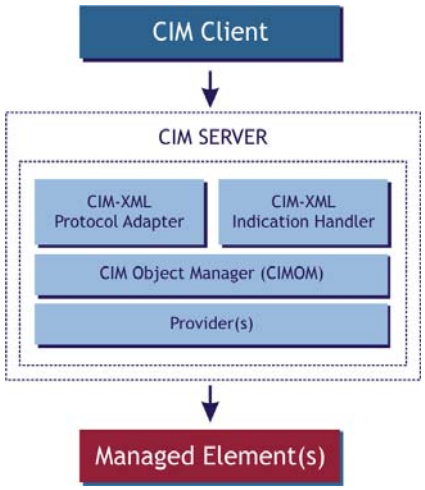


Figura 8 – Componentes de uma infra-estrutura WBEM

O cliente CIM interage com o servidor enviando pedidos (CIM *Operation Message Requests*) e recebendo as respostas (CIM *Operation Message Responses*). O servidor CIM, por seu lado, desempenha um papel de intermediário entre o cliente e os elementos geridos. O núcleo do servidor CIM encontra-se no CIMOM (CIM *Object Manager*) que é a entidade responsável pelo processamento dos pedidos provenientes do(s) cliente(s) e das respostas vindas dos elementos geridos através dos *providers*.

O **Modelo Físico** contém informação relativa ao inventário do sistema em gestão, descrevendo componentes como chassis, cartas, cablagem, etc. Um elemento físico (instância da classe `CIM_PhysicalElement`) representa uma entidade física real. As relações entre componentes (elementos físicos) do sistema são definidas em associações e representam tipicamente a constituição física de um sistema. Note-se que as abstracções presentes neste modelo não representam funcionalidades que os componentes são capazes de realizar. Essas funcionalidades estão representadas na vertente lógica do modelo (subclasses derivadas de `CIM_LogicalElement`). Um elemento físico pode ser classificado em quatro subtipos:

- `CIM_PhysicalPackage`: esta classe descreve "contentores" genéricos com informação sobre gestão, manutenção e reparação. Instâncias desta classe podem também conter outros elementos físicos recorrendo a associações do tipo `CIM_Container`.
- `CIM_PhysicalComponent`: descrição de baixo nível de *hardware* como *chips* ou *physical media*. Também as associações do tipo `CIM_Container` são usadas para descrever os componentes do *Physical Package*.
- `CIM_PhysicalConnector`: classe que descreve os conectores usados para interligar os elementos físicos.
- `CIM_PhysicalLink`: classe usada para descrever a cablagem entre elementos físicos como conectores por exemplo. A associação `CIM_ElementsLinked` indica os elementos físicos que estão ligados.

Como o próprio nome indica, o **Modelo de Sistemas** define as abstracções relacionadas com os **Sistemas** (no sentido lato). Muitos dos conceitos relacionados com um Sistema derivam da classe genérica `CIM_System` do modelo de núcleo. A classe `CIM_System` descreve a agregação de várias partes (componentes) numa única entidade (Sistema).

Além do conceito de sistema em si, este modelo de informação inclui ainda informação sobre os componentes de computação e respectiva funcionalidade, associados à maioria dos sistemas. Inclui-se aqui conceitos do tipo sistemas de ficheiros, sistemas operativos, processos, *threads*, etc. Não existem, contudo, subclasses específicas que descrevam a funcionalidade do sistema (e.g. roteamento, armazenamento, etc.). Essa

informação é descrita pelos serviços disponibilizados ou possíveis de serem fornecidos (a parte lógica do modelo).

O **Modelo de Dispositivos** descreve as funcionalidades disponibilizadas pelo *hardware*, bem como a respectiva configuração e estado. O modelo é bastante abrangente e cobre uma vasta gama de *hardware* desde componentes de baixo nível como sensores, baterias, ventoinhas até dispositivos de alto nível como unidades de armazenamento. Tipicamente, um componente de *hardware* encerra em si várias funcionalidades que são mapeadas no modelo como dispositivos lógicos (`CIM_LogicalDevices`). Os dispositivos físicos são descritos como componentes do **Sistema** (`CIM_System`) que os contém e a relação entre ambos é descrita através de instâncias da classe `CIM_SystemDevice`.

O **Modelo de Métricas** define os componentes de gestão relacionados com as métricas e é baseado em duas classes. Uma para especificar a semântica e uso da métrica (definição) e outra que contém os valores, capturados para uma instância de uma classe de definição.

Num contexto de sistemas de computadores, o termo *policy* é frequentemente usado para descrever configurações do sistema que controlam o comportamento do mesmo (e.g. *security policies*). Em termos genéricos, uma *policy* pode ser definida como um objectivo, um método ou uma linha de execução de acções que guiam e determinam decisões presentes e futuras. O **Modelo de Políticas**, ou *policies*, disponibiliza um *framework* comum para a especificação de comportamentos dos sistemas, sendo suficientemente abstracto para se tornar independente de detalhes específicos de implementação ou de tecnologias mais ou menos complexas. Trata-se, portanto, de um modelo específico para expressar *policies* de uma forma genérica e escalável.

O **Modelo de Utilizador** é um modelo relativo a duas áreas distintas: por um lado, a informação genérica sobre utilizadores (ex.: dados institucionais, informação pessoal) e por outro a gestão de utilizadores de serviços e a respectiva informação de segurança (ex.: autenticação e autorização). Os utilizadores são vistos pelo modelo como entidades genéricas podendo ser vistas como pessoas ou serviços de uma qualquer aplicação.

3.3.2.3 Extensão dos Modelos

A utilização de extensões do modelo permite que fabricantes e/ou outros organismos de desenvolvimento, possam expandir as classes e associações básicas do modelo de modo a

cobrir outras áreas de gestão ou aprofundar com maior nível de detalhe algumas já existentes. A extensão do modelo CIM pode ser feita de diversas maneiras entre as quais:

– Adição de propriedade(s) a classes ou subclasses do modelo CIM ou de esquemas proprietários
– Adição de uma classe ou conjunto de classes ao modelo CIM ou esquema proprietário
– Criação de um esquema e espaço de nomes próprio

Algumas das modificações possíveis de se efectuar sobre um esquema são:

– Uma classe pode ser adicionada ou removida do esquema
– Uma propriedade pode ser adicionada ou removida de uma classe
– Uma classe pode ser adicionada como uma sub ou superclasse de outras já existentes
– Uma classe pode-se transformar numa associação bastando para tal acrescentar um qualificador de associação e duas ou mais referências
– Um qualificador pode ser adicionado ou removido de um elemento do esquema
– Um método pode ser adicionado a uma classe
– Um método novo pode-se sobrepor a um método herdado

Na prática, o desenvolvimento de extensões do modelo significa, na maioria dos casos, a implementação de subclasses das classes já existentes no núcleo e nos modelos comuns, obedecendo assim ao princípio já descrito de que com as extensões pretende-se refinar, aumentando o nível de detalhe, áreas de gestão específicas.

O projecto de um novo esquema é uma tarefa algo complexa pois levanta uma série de perguntas:

– Em que esquema se deve basear?
– A aplicação ou dispositivo “encaixa” no novo esquema?
– Quais são as associações e dependências?
– Que cenários se poderão colocar no futuro face a este esquema?

Independentemente das respostas e da complexidade do novo esquema, há dois factores que são absolutamente essenciais: **Eficiência** e **Usabilidade**.

No desenvolvimento de um novo esquema, dever-se-á ter em conta que se está a criar uma linguagem que será usada para comunicar uma estrutura de coisas (elementos lógicos e/ou físicos) num sistema de informação. Muito provavelmente, esta estrutura crescerá ao longo do tempo e será interpretada por vários utilizadores. Dada a natureza dos esquemas, dificilmente se conseguirá definir significados absolutos para toda a estrutura. No entanto, no desenvolvimento do esquema será razoável procurar um equilíbrio entre o nível de detalhe e o nível de compreensão. Embora seja desejável ter o maior nível de detalhe possível, a componente complexidade terá forçosamente que ser equacionada.

3.3.3 CIM Namespace

Um espaço de nomes CIM (vulgarmente designado por *namespace*) não é mais do que um agrupamento lógico de classes e instâncias que permite limitar a visibilidade e o controlo das mesmas. Um espaço de nomes não é uma localização física mas pode ser visto como bases de dados lógicas contendo classes e instâncias específicas.

Um dos requisitos do modelo CIM é a necessidade de identificar única e inequivocamente instâncias de classes. Para tal, cada classe tem uma ou mais propriedades chave que, em conjunto, constituem um identificador único (semelhante a uma chave primária numa base de dados).

Dado que o modelo CIM permite várias implementações e extensões ao modelo, não é suficiente pensar num objecto como uma combinação das suas propriedades chave. O nome deve também identificar a implementação onde está "alojado" o objecto. Assim, o nome de um objecto consiste num caminho de espaço de nomes (*Namespace Path*) e no caminho do modelo (*Model Path*). Enquanto que o primeiro indica qual a implementação CIM a usar, o segundo permite navegar no esquema CIM relativo à implementação seleccionada. O caminho do modelo é a concatenação do nome da classe com as respectivas propriedades chave. A figura seguinte ilustra os componentes do nome de um objecto: `root/cimV2:CIM_Disk.key1=value1`.

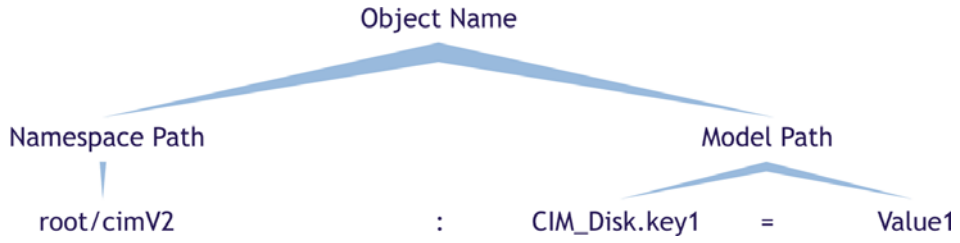


Figura 9 – Composição do nome de um objecto

O *namespace path* identifica um *namespace* numa implementação CIM enquanto que o *model path* é usado para descrever o caminho para um objecto em particular ou para identificar um objecto em particular num dado *namespace*. Note-se que dentro do mesmo *namespace*, o nome de uma classe deve ser único, podendo, no entanto, esse nome ser repetido num outro *namespace*.

3.4 Web-Based Enterprise Management

Um dos objectivos do grupo DMTF é promover a interoperabilidade entre diversas soluções de gestão. A iniciativa *Web-Based Enterprise Management* (WBEM) é um conjunto

de tecnologias desenvolvidas para uniformizar a gestão de ambientes computacionais de uma forma *standard* e não proprietária. Contudo, inicialmente não estava definida nenhuma tecnologia para a troca ou transferência de dados CIM entre diferentes plataformas. Deste modo, acabaram por surgir diferentes implementações WBEM que, apesar de assentarem na mesma base ou seja, o modelo CIM, não comunicavam entre si dado não haver uma forma *standard* para o fazer. Por exemplo, uma ferramenta de gestão como o *WMI CIM Studio for Windows* não consegue gerir ou consultar informação de gestão de um sistema *Unix* ou *MacOS*. Existem contudo outras ferramentas e projectos que implementam a infra-estrutura WBEM de acordo com as especificações do DMTF, tornando-se assim soluções universais. Um exemplo é o *Open Pegasus* [42]. O *Open Pegasus* é uma implementação *open-source* publicada e suportada por um grupo independente de tecnologias e fabricantes de seu nome *The Open Group* [41]. Dado que actualmente o *Open Pegasus* é, talvez, um dos projectos mais completos em termos das tecnologias abordadas, no Anexo A é descrito com maior detalhe.

Em Janeiro de 2003 o DMTF publicou as primeiras versões finais de *standards* para a troca de dados CIM entre diferentes implementações WBEM. Para que a comunicação de informação de gestão CIM ocorra entre plataformas diferentes houve dois problemas que tiveram de ser ultrapassados. Em primeiro lugar é necessário empacotar a informação de gestão de uma forma universal, independente do tipo de plataforma ou sistema operativo. Para este efeito o DMTF escolheu a linguagem *eXtensible Markup Language* (XML) [30]. Dado que existem várias formas de representar a informação CIM em XML, foi definido um *standard* - **Representação de CIM em XML** [31] para assegurar que todos os intervenientes o façam da mesma maneira.

Por outro lado, a informação tem que ser transportada entre as entidades intervenientes no processo de comunicação de forma *standard*. O DMTF escolheu um dos protocolos mais amplamente usados hoje em dia, o *Hyper Text Transfer Protocol* (HTTP) [32][33]. O HTTP foi escolhido dado que, pela sua larga utilização nas redes actuais, os dispositivos de rede estão preparados para um tratamento capaz e eficiente deste protocolo. Ele não é, normalmente, limitado por *firewalls* o que facilita bastante a troca de informação de gestão entre entidades. Além disso, a segurança no transporte dos dados pode ser assegurada recorrendo ao transporte seguro de HTTP: o HTTPS [34].

Agregando todos os blocos, temos que o WBEM é composto por:

– **Modelo de Informação Comum** (CIM), permite o uso de um formato e linguagem comuns na recolha e descrição dos dados de gestão

– **Representação de CIM em XML,** define elementos XML escritos em DTD (*Document Type Definition*) [36] que são usados para representar as classes CIM e suas instâncias.

– **Operações CIM sobre HTTP**, define o capeamento das operações CIM em HTTP o que permite que implementações do modelo CIM interoperem num espaço aberto e estandardizado.

O modelo de informação comum CIM foi já descrito na secção anterior pelo que de seguida serão introduzidos os restantes dois blocos do WBEM: representação de CIM em XML e as operações CIM sobre HTTP.

3.4.1 Representação de CIM em XML

Existem várias maneiras de representar a informação CIM em XML pelo que se tornou necessário definir um formato normalizado. Antes de descrever o processo de normalização levada a cabo pelo DMTF, convém introduzir umas pequenas notas sobre XML, mais concretamente DTD's.

XML [30] é uma linguagem de *markup* ou seja, utiliza marcas para anotar e organizar a informação de determinada maneira. Um documento XML é uma colecção de dados representados nessa linguagem. Uma das grandes vantagens do XML é a sua flexibilidade dado que as marcas usadas não são previamente impostas o que permite representar a informação de múltiplas formas. No entanto, quando a informação necessita ser trocada entre várias entidades, torna-se necessário assegurar a consistência dos dados. O que torna um documento diferente de um amontoado de caracteres é a sua consistência. Um dos métodos usados em XML para assegurar a referida consistência é o uso de *Document Type Definition* (DTD) [36].

Uma DTD, na sua versão mais simplista, é um conjunto de regras que definem os elementos e respectivos atributos para um documento XML. Pode-se dizer que uma DTD define uma gramática para um documento XML pois indica às aplicações e utilizadores o que significa cada elemento e como poderá ser usado. Ou seja, quando se usa uma DTD, sujeita-se o documento XML a um conjunto de regras que ditam a forma como ele é escrito (*markup*). As DTD's consistem essencialmente em declarações de elementos e seus atributos. Tipicamente, as DTD's são usadas quando:

- É necessário criar e gerir um grande número de documentos (a DTD permite criar regras que todos os documentos deverão seguir)
- É necessário definir claramente qual o tipo de escrita (*markup*) que pode ser usado em determinados documentos e a sequenciação da mesma.
- Existem documentos que vários utilizadores partilham pelo que é necessário uma referência ou base de trabalho comum.

Para o mapeamento de CIM em XML, o DMTF tinha pela frente duas possibilidades:

1) **O mapeamento do esquema** em que uma DTD é usada para descrever as classes CIM, sendo as instâncias CIM mapeadas em documentos XML válidos segundo a DTD definida. No essencial significa que cada classe CIM gera o seu próprio fragmento DTD ou seja, os nomes dos elementos XML são obtidos directamente dos correspondentes elementos CIM. Esta opção é referida pelo DMTF como sendo um *schema* XML [59]. Não confundir com a norma XML *schema* do W3C [67].
2) **O mapeamento do *metaschema*** consiste no uso de uma DTD para descrever o *metaschema* CIM, sendo neste caso ambas as classes e instâncias CIM mapeadas em documentos XML válidos segundo essa DTD. Por outras palavras, uma DTD é usada para descrever, de um modo genérico, uma classe ou instância CIM. Os nomes dos elementos CIM são mapeados em atributos e valores de elementos XML em vez de nomes de elementos XML.

Apesar de existirem benefícios óbvios na primeira opção (validação mais eficaz, representação mais intuitiva de CIM em XML), o DMTF optou pelo segundo modelo pelas seguintes razões:

- É apenas necessário um *metaschema* DTD *standard* em vez de um vasto número de DTD's o que reduz a complexidade de gestão e administração dos mapeamentos.
- Uma DTD XML obriga à ordenação da lista de elementos. Num mapeamento estático significaria fixar uma ordem para os vários elementos (propriedades, métodos, qualificadores) ou definir um conjunto alargado de mapeamentos que tivesse em conta as várias ordenações possíveis para os elementos sendo que neste caso, o número de DTD's cresceria exponencialmente com o número de elementos da lista.
- Num mapeamento em *schema* XML, os elementos do esquema CIM (classe, propriedade, qualificador, método) são mapeados em elementos XML. No entanto, o mapeamento no *schema* XML requer uma maior granularidade ao nível das propriedades o que tornaria o processo de mapeamento extremamente complexo.
- O mapeamento do *metaschema* apenas introduz um pequeno e fixo número de termos em XML: classe, instância, propriedade, etc.
- Apesar de um mapeamento em *schema* XML permitir a validação de instâncias nas respectivas classes, tal só seria possível se a hierarquia de classes fosse completamente plana. O facto de não o ser faz com que propriedades herdadas de classes de nível superior sejam escondidas da DTD, podendo causar falhas na validação de determinadas instâncias.

A gramática XML, definida pela DTD publicada em [59], pode ser usada quer para a representação de declarações CIM (classes, instâncias e qualificadores) quer para mensagens CIM (para posterior transporte em HTTP como veremos na secção seguinte).

Para não se tornar fastidioso, a gramática definida pela DTD não será aqui detalhada, pelo que serão apresentados apenas dois exemplos para dar uma ideia de como são definidas, por um lado, as classes CIM com a declaração da classe `CIM_LogicalPort` e, por outro lado, as mensagens CIM, sendo exemplificada uma mensagem representando um pedido de uma instância da classe `WIN32_Process`.

A figura 10 apresenta um extracto da declaração da classe `CIM_LogicalPort` em XML, válida de acordo com o *metaschema* definido pelo DMTF.

```
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.1.1">
  <CLASS NAME="CIM_LogicalPort" SUPERCLASS="CIM_LogicalDevice">
    <QUALIFIER TRANSLATABLE="true" NAME="Description" TYPE="string">
       <VALUE>
          The abstraction of a port or connection point of a Device. This
          object ... independently of the EthernetAdapter.
       </VALUE>
    </QUALIFIER>
    <PROPERTY NAME="Speed" TYPE="uint64">
       <QUALIFIER TRANSLATABLE="true" NAME="Description" TYPE="string">
          <VALUE>The speed of the Port in Bits per Second.</VALUE>
       </QUALIFIER>
       <QUALIFIER TRANSLATABLE="true" NAME="Units" TYPE="string">
          <VALUE>Bits per Second</VALUE>
       </QUALIFIER>
    </PROPERTY>
    <PROPERTY NAME="MaxSpeed" TYPE="uint64">
       <QUALIFIER TRANSLATABLE="true" NAME="Description" TYPE="string">
          <VALUE>The max speed of the Port in Bits per Second.</VALUE>
       </QUALIFIER>
       <QUALIFIER TRANSLATABLE="true" NAME="Units" TYPE="string">
          <VALUE>Bits per Second</VALUE>
       </QUALIFIER>
    </PROPERTY>
    ...
  </CLASS>
</CIM>
```

Figura 10 – Declaração de uma classe CIM em XML

Tal como em qualquer documento XML bem formado, no prólogo é indicada a versão do XML e o tipo de *encoding* para o conjunto de caracteres usado no conteúdo. Quanto ao elemento raiz, CIM, trata-se da versão 2.0 tendo sido usada a versão 2.1.1 na DTD

que valida o documento. A declaração `CLASS` indica a classe a declarar (`CIM_LogicalPort`) e a superclasse (`CIM_LogicalDevice`) de que deriva a classe declarada. Logo a seguir vem um qualificador da classe (`Description`) cujo conteúdo é uma descrição sumária da mesma. Depois do qualificador da classe, seguem-se as propriedades da mesma. Para simplificar o exemplo, foram apresentadas apenas duas propriedades da classe: `Speed` e `MaxSpeed`. Cada propriedade tem os seus respectivos qualificadores.

As mensagens CIM são definidas para permitirem a troca de informação, sobre um protocolo de transporte, entre duas ou mais entidades (máquinas), normalmente um servidor e um cliente. Quanto à sua representação em XML, apresenta-se de seguida um exemplo na figura 11.

```
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.1.1">
  <MESSAGE ID="12345" PROTOCOLVERSION="1.0">
    <SIMPLEREQ>
      <IMETHODCALL NAME="GetInstance">
        <LOCALNAMESPACEPATH>
          <NAMESPACE NAME="root"/>
          <NAMESPACE NAME="CIMV2"/>
        </LOCALNAMESPACEPATH>

        <IPARAMVALUE NAME="InstanceName">
          <INSTANCENAME CLASSNAME="Win32_Process">
            <KEYBINDING NAME="Handle">
              <KEYVALUE>44</KEYVALUE>
            </KEYBINDING>
          </INSTANCENAME>
        </IPARAMVALUE>

        <IPARAMVALUE NAME="LocalOnly">
          <VALUE>FALSE</VALUE>
        </IPARAMVALUE>

      </IMETHODCALL>
    </SIMPLEREQ>
  </MESSAGE>
</CIM>
```

Figura 11 – Mensagem CIM em XML

O exemplo apresentado na figura 11 representa um pedido de um cliente para que o servidor CIM lhe retorne a instância da classe `Win32_Process` cuja propriedade `Handle` seja igual a 44.

Tal como na declaração de classe, as primeiras duas linhas incluem a versão da especificação XML, o tipo de *encoding*, a versão do modelo CIM e a versão da DTD. O `ID` da mensagem identifica unicamente a mensagem em causa (é usado pelas entidades em comunicação para distinguir que respostas correspondem a que pedidos) e o atributo `PROTOCOLVERSION` define a versão do mapeamento CIM usado na mensagem. A seguir é declarado o tipo de mensagem `<SIMPLEREQ>` para um pedido simples (o outro tipo possível é o `<MULTIREQ>` para pedidos múltiplos na mesma mensagem). Segue-se a declaração do método a invocar `GetInstance` e a *path* do *namespace* CIM, `root/CIMV2`. O passo seguinte é a passagem de parâmetros ao método *GetInstance*: o nome da classe (`Win32_Process`) e o valor de uma propriedade chave que permita ao CIMOM retornar apenas a instância desejada. A passagem do parâmetro `LocalOnly` com o valor de `FALSE` faz com que seja retornada todas as propriedades e respectivos valores mesmo que tenham sido herdadas de classes superiores (por defeito este valor é `TRUE`).

No Anexo B, é apresentado um exemplo mais completo de comunicação entre duas entidades CIM sendo analisada apenas a troca de informação ao nível de CIM/XML.

3.4.2 Operações CIM sobre HTTP

A iniciativa WBEM define um conjunto de operações que um cliente WBEM pode implementar através da norma [38]. Todas as operações provenientes de um cliente são definidas como invocações de um ou mais métodos. Os métodos podem ser intrínsecos ou extrínsecos caso sejam feitos sobre um *namespace* CIM ou sobre uma classe CIM estática ou instância de um classe CIM.

O conjunto de operações organiza-se nas seguintes categorias:

- **Leitura**

  GetClass – usada para retornar uma classe do *namespace* alvo

  EnumerateClasses – usada para enumerar subclasses de uma classe CIM do *namespace* alvo

  EnumerateClassNames – usada para enumerar nomes de subclasses de uma classe CIM no *namespace* alvo

  GetInstance – usada para retornar uma instância CIM do *namespace* alvo

  EnumerateInstances - usada para enumerar instâncias de uma classe CIM do *namespace* alvo

  EnumerateInstanceNames – usada para enumerar nomes de instâncias de uma classe CIM no *namespace* alvo

  GetProperty – usada para retornar o valor de uma única propriedade de uma instância CIM no *namespace* alvo

- **Escrita**

  SetProperty – usada para o *set* do valor de uma propriedade numa instância CIM do *namespace* alvo

- **Manipulação de instâncias**

  CreateInstance – usada para criar uma instância CIM no *namespace* alvo

  ModifyInstance – usada para modificar uma instância CIM no *namespace* alvo

  DeleteInstance – usada para eliminar uma instância CIM existente no *namespace* alvo

- **Manipulação do *schema***

  CreateClass – usada para criar uma classe CIM no *namespace* alvo

  ModifyClass – usada para modificar uma classe CIM existente no *namespace* alvo

  DeleteClass – usada para eliminar uma classe CIM existente no *namespace* alvo

- **Associações**

  Associators – usada para enumerar objectos CIM (classes ou instâncias) associados a um determinado objecto CIM

  AssociatorNames – usada para enumerar nomes de objectos CIM (classes ou instâncias) associados a um determinado objecto CIM

  References – usada para enumerar objectos do tipo associação referentes a um determinado objecto CIM (classe ou instância)

  ReferenceNames – usada para enumerar nomes de objectos do tipo associação referentes a um determinado objecto CIM (classe ou instância)

- ***Query***

  ExecQuery – usada para executar uma *query* sobre um *namespace* alvo

- **Qualificadores**

  GetQualifier – usada para retornar um *qualifier* do *namespace* alvo

  SetQualifier – usada para o *set* de um *qualifier* do *namespace* alvo

  DeleteQualifier – usada para eliminar um *qualifier* do *namespace* alvo

  EnumerateQualifiers – usada para enumerar *qualifiers* do *namespace* alvo

A especificação de transporte de CIM em HTTP define as trocas de informação entre entidades CIM como mensagens CIM. Uma mensagem CIM é um conjunto de dados com um pedido ou uma resposta usado para a troca de informação entre entidades CIM. As mensagens CIM podem ser de dois tipos:

– *CIM Operation Message*, usada para invocar uma operação (qualquer uma das operações atrás descrita) no *namespace* alvo

– *CIM Export Message*, usada para comunicar informação acerca de um *namespace* CIM (é uma mensagem puramente informativa e não define qualquer tipo de operação sobre o *namespace* alvo)

O *standard* especifica também que qualquer tentativa para envio de dados deverá usar o método `M-POST` inicialmente. A utilização do método `M-POST` deriva do uso da norma "HTTP Extension Framework" [39] que permite o uso de *headers* adicionais no cabeçalho HTTP. Considere-se o seguinte exemplo, onde é analisado apenas o cabeçalho HTTP numa mensagem trocada entre duas entidades CIM.

```
M-POST /cimom HTTP/1.1
HOST: www.fe.up.pt
Content-Type: application/xml; charset="utf-8"
Content-Length: 500
Man: http://www.dmtf.org/cim/mapping/http/v1.0; ns=44
44-CIMOperation:MethodCall
44-CIMMethod: GetInstance
44-CIMObject: root%2fcimv2
```

O *header* começa com a indicação de que se trata de um pedido feito através do método `M-POST`, usando a versão 1.1 do protocolo HTTP. A declaração `/cimom` serve apenas para indicar ao CIM *Object Manager* que deverá proceder ao processamento do *payload*. A linha `HOST` identifica o endereço do `CIMOM`. O `Content-Type` indica o tipo de dados presente no *payload* (neste caso XML) e o *charset* usado. O `Content-Length` indica o tamanho (em *bytes*) do *payload*.

Tratando de um pedido `M-POST`, as extensões do *header* devem ser declaradas através do campo `Man` (diminutivo de *mandatory*) que indica ao `CIMOM` qual o *name space* HTTP a que se refere). As mensagens CIM trocadas entre entidades deverão sempre conter esta linha. O valor de `ns` (abreviatura de *name space*) [39] correspondente a um número de dois dígitos, aleatório, é usado pelos campos do *header* seguintes para os identificar como pertencendo à declaração da extensão. De seguida é especificada a operação CIM como sendo do tipo *MethodCall* (o outro tipo possível seria *MethodResponse*). Resta a declaração do método – *GetInstance* e o *namespace* sobre o qual será invocado. Note-se o uso de %2f que corresponde ao código ASCII do caracter "/" de acordo com [40].

3.5 Windows Management Interface - WMI

Apesar de o DMTF ter sido fundado em 1992 e de os seus esforços terem produzido alguns *standards* como o CIM ou o WBEM, as suas iniciativas tem ainda uma reduzida implementação nos equipamentos e sistemas operativos actuais.

Windows Management Interface (WMI) é a designação dada pela Microsoft à sua implementação da iniciativa WBEM nos seus sistemas operativos. O WMI foi introduzido pela Microsoft, inicialmente em alguns dos seus sistemas operativos, com o objectivo de suportar a gestão de sistemas numa organização, sendo actualmente implementado em todos os sistemas operativos da Microsoft. Esta característica dos sistemas operativos Microsoft, aliada a uma larga implementação dos mesmos nos parques informáticos actuais, fez com que o WMI fosse escolhido como a principal tecnologia de desenvolvimento da plataforma de gestão.

Trata-se de uma tecnologia que permite a gestão remota de sistemas *Windows* e respectivas aplicações. Através do WMI é possível desenvolver ferramentas simples para a recolha de informação de gestão ou para alteração de configurações em máquinas remotas. É uma enorme quantidade de dados que podem ser recolhidos (*hardware*, *software*, performance, *drivers*, BIOS, etc.) pois o WMI recolhe esta informação a partir de uma diversidade de API's.

Colmatando a falta de decisões no que respeita à troca de informação de gestão CIM, a tecnologia usada em WMI permite que a informação seja exposta, usando a mesma API, quer local quer remotamente. Para tal, o WMI usa a tecnologia *Component Object Model* (COM/*Distributed* COM) [44] para o acesso à informação. Esta acaba por ser uma desvantagem em relação à escolha efectuada. Apesar da organização da informação e o modo como ela é apresentada respeitarem os *standards* definidos para o efeito, mais concretamente o modelo CIM, o modo como a informação é transportada assenta numa tecnologia fechada o que impossibilita a gestão de sistemas operativos de outros fabricantes.

Para abreviar este problema, várias soluções poderão ser estudadas. No entanto, existem duas que se afiguram, provavelmente, como as mais simples e eficazes. Uma delas implica desenvolvimentos adicionais ao nível da plataforma de gestão. Esta teria que ser capaz de identificar em primeiro lugar qual o sistema operativo da máquina gerida e só então depois partir para a gestão da mesma usando as tecnologias apropriadas. Esta solução, apesar de transparente para o administrador e utilizador final da plataforma implica um esforço maior ao nível do desenvolvimento da plataforma e a introdução de alguma lógica para a identificação do sistema operativo. Uma outra solução possível seria o desenvolvimento e instalação de *software* cliente nas máquinas a gerir, *software* esse que seria instalado em máquinas onde os *standards* de acesso e transporte à informação CIM (Representação de CIM em XML - **3.4.1** e Operações CIM sobre HTTP - **3.4.2**) não fossem utilizados. A desvantagem óbvia desta solução é o aumento de tarefas "logísticas" da parte do administrador já que teria que instalar o referido *software* eventualmente em várias máquinas. Além disso, tendo em conta a realidade actual e o uso alargado de sistemas operativos da Microsoft nas instituições, seria extremamente complicado distribuir e manter *software* cliente em todas as máquinas com esse tipo de sistemas operativos.

3.5.1 Arquitectura

Em termos arquitectónicos, o WMI pode ser decomposto em três secções. Por um lado temos as aplicações ou clientes de gestão que comunicam directamente com o CIMOM (CIM *Object Manager*). O CIMOM, juntamente com o repositório de objectos CIM constitui o núcleo do WMI. Nas plataformas *Windows*, o serviço responsável por esta tarefa é o `winmgmt.exe`. Além de representar a interface de comunicação com as aplicações de gestão, o CIMOM é também responsável por obter a informação necessária através dos *providers*. Estes realizam a interface entre o CIMOM e os dados de gestão das várias áreas. A figura seguinte representa a arquitectura do WMI.

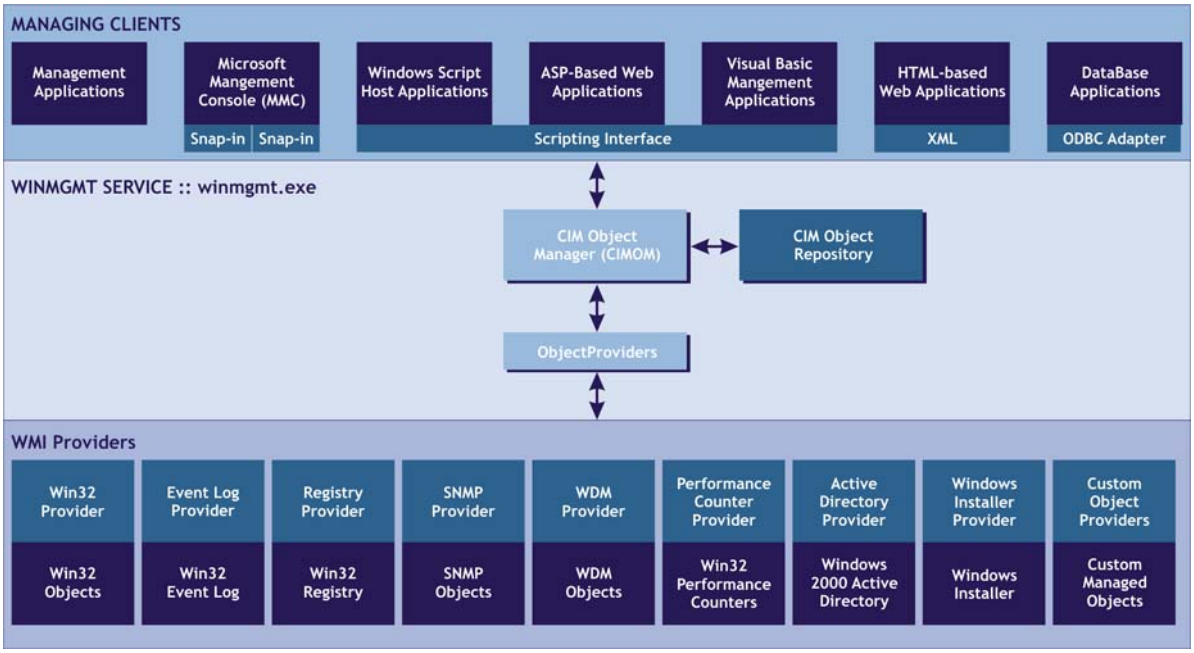


Figura 12 – Arquitectura WMI (fonte [44])

Os clientes de gestão (aplicações de gestão) interagem com o serviço de gestão através de uma *Application Programming Interface* (API). Apesar de actualmente já existirem *standards* para esta comunicação, Operações CIM sobre HTTP (secção **3.4.2**) e Representação de CIM em XML (secção **3.4.1**), na altura em que a Microsoft desenvolveu e implementou o WMI, os esforços da DMTF encontravam-se ainda numa fase inicial pelo que a opção foi o uso de interfaces COM/DCOM [44]. As interfaces COM/DCOM permitem que o desenvolvimento de aplicações de gestão possa ser feito em qualquer linguagem que suportem o acesso às mesmas. Linguagens como o *Visual Basic*, *VBScript*, *Windows Scripting Host*, ASP, Perl ou PHP são exemplos de linguagens que podem ser usadas para o desenvolvimento de aplicações de gestão.

Além da definição das interfaces de acesso, o CIMOM necessita de um espaço de armazenamento que lhe permita gerir o *metaschema* ou seja, as definições das classes CIM.

Esta informação, bem como outros dados estáticos manipulados pelo CIMOM (e.g. configurações de segurança no acesso ao serviço), é mantida num repositório designado por CIM *Object Repository*.

A disponibilização da informação dinâmica, correspondente aos dados de gestão do modelo CIM, é feita através de *providers*. Na realidade, estes *providers* não são mais do que servidores COM que suportam um conjunto bem definido de interfaces e que podem ser consultados pelo CIMOM ou chamar o CIMOM para o informar de alterações em determinados dados.

Como foi já referido na secção **3.3.2.3,** as extensões do modelo CIM permitem refinar a gestão, cobrindo com mais detalhe e disponibilizando outras funcionalidades para áreas de gestão específicas. Em WMI, foram criadas extensões que permitem gerir de forma mais completa os sistemas operativos *Windows*. O WMI define classes e/ou propriedades que fornecem um maior detalhe e informação mais específica sobre o ambiente gerido. A convenção usada pela Microsoft foi a de prefixar com `Win32_` qualquer classe derivada do modelo CIM. Por exemplo, a classe `Win32_NetworkAdapter` tem como superclasse a classe `CIM_NetworkAdpater` do modelo CIM. A disponibilização dos dados de gestão, quando requisitados, e a gestão das classes suportadas é da responsabilidade do CIMOM.

3.5.2 Espaços de Nomes

Assente no modelo CIM, o WMI organiza e agrupa as várias classes em unidades lógicas (colecções) designadas por *namespaces* (ver secção **3.3.3**). A sua organização é semelhante à estrutura de directórios num disco sendo que as classes podem ser vistas como os ficheiros dentro desses directórios. A figura 13 ilustra esta analogia.

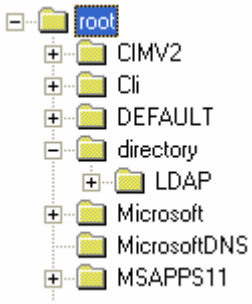


Figura 13 – Espaços de nomes em WMI

A localização de um *namespace* é descrita por um caminho (*path*) e em cada um dos *namespaces* podemos encontrar a sua colecção de classes e instâncias. Para identificar unicamente uma classe é necessário indicar também o seu *namespace*, por exemplo: `root/cimV2:CIM_Processor`.

A descrição de um *namespace* e das suas classes introduz o conceito de caminho do objecto (*Object Path*). O caminho do objecto é uma forma de especificar uma máquina, um *namespace*, uma classe ou uma instância.

Por exemplo, `\\dragom.fe.up.pt\root\cimV2:Win32_Process` refere-se à classe `Win32_Process` no *namespace* `root\cimV2` na máquina `\\dragom.fe.up.pt`. Por omissão é sempre considerada a máquina corrente e o *namespace* em uso. Para especificar instâncias de uma classe basta indicar as propriedades chave (semelhante a uma chave primária) e os seus valores:

```
\\dragom.fe.up.pt\root\cimV2:Win32_Process.Handle=”728”
```

No caso de não ser especificada nenhuma propriedade chave, o WMI retornará todas as instâncias da classe requisitada. Para classes que não possuem qualquer propriedade chave (*Singleton Classes*) e em que existe apenas uma e uma só instância, o caminho do objecto contém o sufixo `=@` a seguir ao nome da classe:

```
\\dragom.fe.up.pt\root\cimV2:SingletonTest=@
```

3.5.3 Considerações sobre Segurança

Dado que o WMI é um serviço, ele é executado sob o utilizador *Local System*, o qual, virtualmente, tem um controlo ilimitado sobre o computador. Quando um cliente faz um pedido ao WMI, é o serviço que se encarrega de chamar as API’s respectivas para recolher a informação. No entanto, isto não significa que a segurança inerente aos sistemas *Windows* tenha sido ladeada. Na realidade, sempre que é feita uma conexão ao serviço WMI, são passadas as credenciais do utilizador através da infra-estrutura DCOM. O WMI irá depois fazer um *impersonate* para realizar os pedidos. Esta técnica de *impersonation* consiste na passagem das credenciais dos utilizadores que pretendem aceder à informação disponibilizada pelo serviço como se o serviço WMI corresse sob o utilizador cujas credenciais foram passadas. Desta forma, assegura-se que qualquer utilizador executa apenas as operações para as quais tem permissões.

Para ligações a máquinas remotas, o WMI permite que sejam passadas as credenciais de qualquer utilizador com permissões para o efeito. Por omissão, as credenciais utilizadas pelo serviço serão sempre as do utilizador corrente. O exemplo de ligação ao serviço exposto anteriormente pode ser transformado no seguinte:

```
$locator = new COM("WbemScripting.SWbemLocator");
$wbemServ = $locator->ConnectServer($host,$namespace,$user,$pass);
```

A configuração do nível de segurança no acesso ao WMI é suportada no esquema de segurança dos sistemas *Windows*. Esta funcionalidade é bastante útil para definir restrições para ligações remotas. Estas configurações podem ser definidas no *snap-in Computer Management* em *WMI Control* conforme a figura 14.

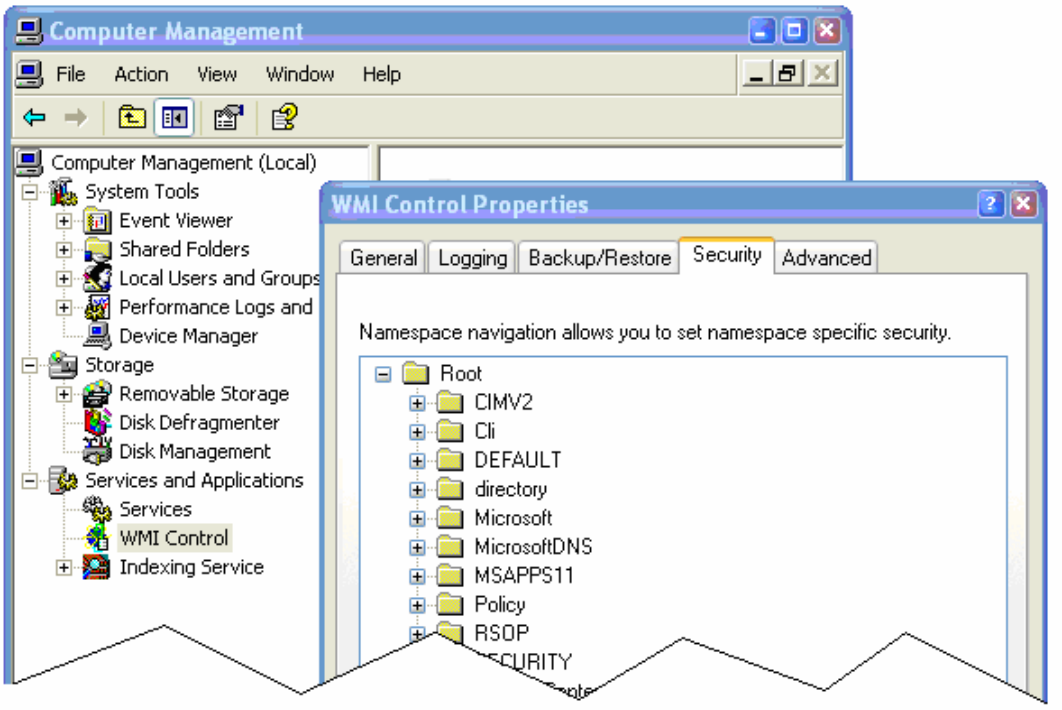


Figura 14 – Configuração de segurança no acesso ao WMI

3.6 Simple Network Management Protocol – SNMP

Desde a sua criação em 1988 que o *Simple Network Management Protocol* (SNMP) [8] pretendia ser uma solução a curto prazo para a gestão de redes. Porém, esta solução de curto prazo enraizou-se e com as actualizações de 1993 com a versão 2 [9][10][11] e em 1998 com a versão 3 [12][13] fez com que se mantivesse como o principal protocolo para a gestão de redes IP (e em alguns casos, também de sistemas). De entre várias características que o tornaram popular ao longo dos anos salienta-se a sua flexibilidade e um baixo grau de exigência de recursos de processamento. O SNMP é baseado no modelo agente/servidor e consiste num gestor, um agente, uma base de dados com informação de gestão, os objectos geridos e um protocolo de rede (figura 15).

Figura 15 – Modelo SNMP de gestão

O gestor representa a interface entre o administrador (humano) e o sistema de gestão. O agente representa a interface entre o gestor e os dispositivos geridos (equipamentos). O gestor e o agente usam uma base de dados de informação de gestão – *Management Information Base* (MIB) e um conjunto relativamente pequeno de comandos para trocar a informação de gestão.

3.6.1 Arquitectura

O SNMP foi desenvolvido como um protocolo do nível aplicacional na suite TCP/IP, correndo sobre *User Datagram Protocol* (UDP). Quer isto dizer que não é orientado às conexões ou seja, qualquer mensagem trocada entre entidades SNMP é considerada como uma transacção única, não havendo lugar para qualquer tipo de confirmação na recepção das mesmas. A forma como a informação de gestão é partilhada entre o agente e a estação gestora encontra-se apresentada na figura 16.

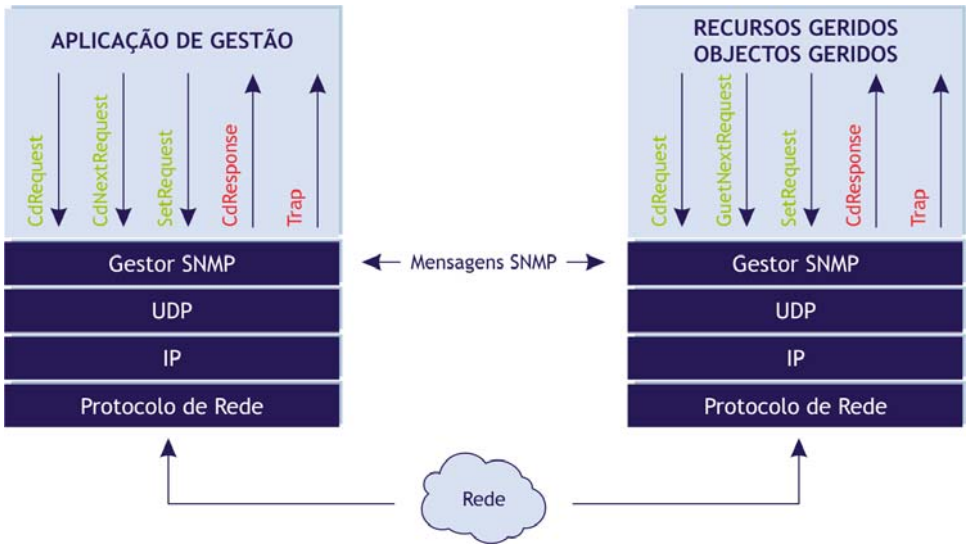


Figura 16 – Arquitectura do SNMP (fonte [46])

Existem cinco tipos básicos de mensagens: GET, GET-NEXT, GET-RESPONSE, SET e TRAP. Os dois primeiros permitem informação para uma variável específica. A mensagem GET-RESPONSE será a resposta do agente com a informação solicitada. Uma mensagem de SET permite ao gestor escrever determinados valores em certas variáveis. A

mensagem TRAP é gerada autonomamente pelo agente e permite que este informe, espontaneamente o gestor sobre a ocorrência de determinados eventos.

Além das mensagens trocadas entre entidades SNMP, existe a questão do conteúdo ou seja, a informação de gestão. Neste campo, o SNMP faz uso da *Management Information Base* (MIB), a qual está organizada segundo uma estrutura em árvore onde as variáveis são representadas pelas folhas dos ramos da árvore. Cada folha da árvore representa um recurso gerido e é identificado por uma etiqueta numérica designada por *Object Identifier* (OID). Na figura 17 está representada parte da estrutura em árvore, com maior incidência na *Host Resource* MIB [47] da MIB2 [48].

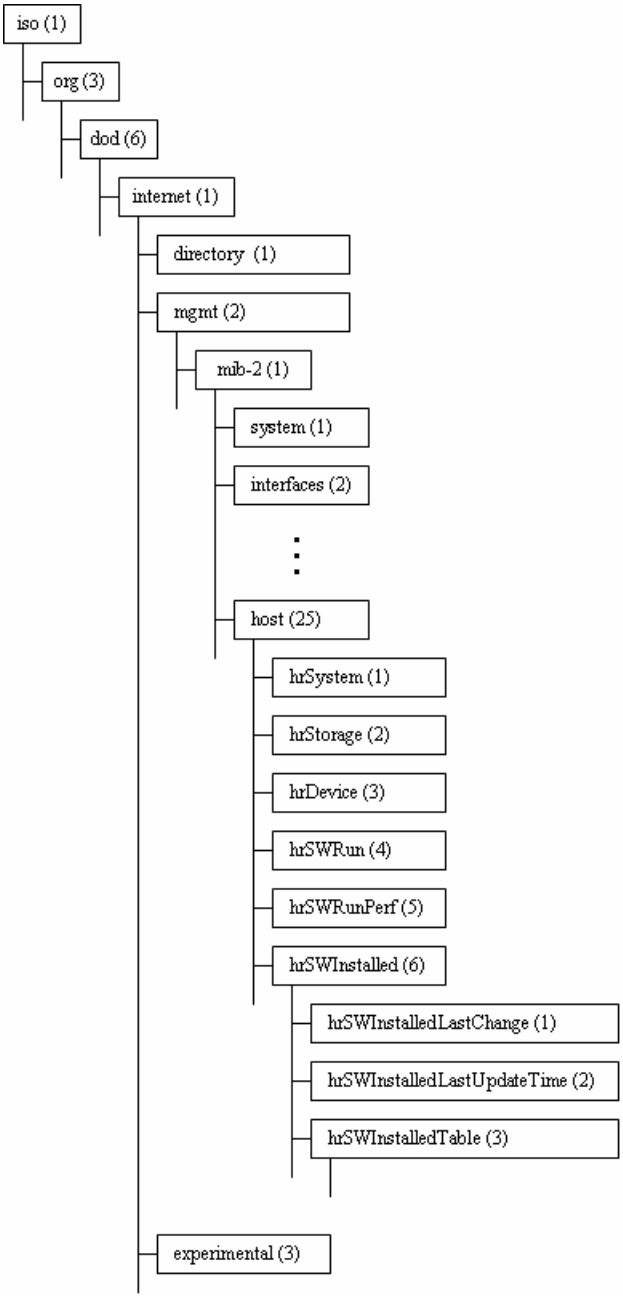


Figura 17 – Estrutura em árvore da MIB

3.6.2 Mensagens SNMP

Como foi já dito, a versão inicial do SNMP tem vindo gradualmente a ser melhorada pelo que existe já a versão 3 do protocolo. No entanto, e dado tratar-se de um protocolo desenvolvido especialmente para a gestão de redes, os fabricantes de sistemas operativos optam por não implementar a versão 3 nos seus sistemas. Ao nível da plataforma de gestão desenvolvida, é utilizada a versão 2 do protocolo dado que é a versão suportada pela maioria dos sistemas operativos. Assim, as mensagens trocadas entre entidades SNMP respeitam os formatos definidas pela respectiva versão (SNMPv2), sendo a sua estrutura representada na figura 18.

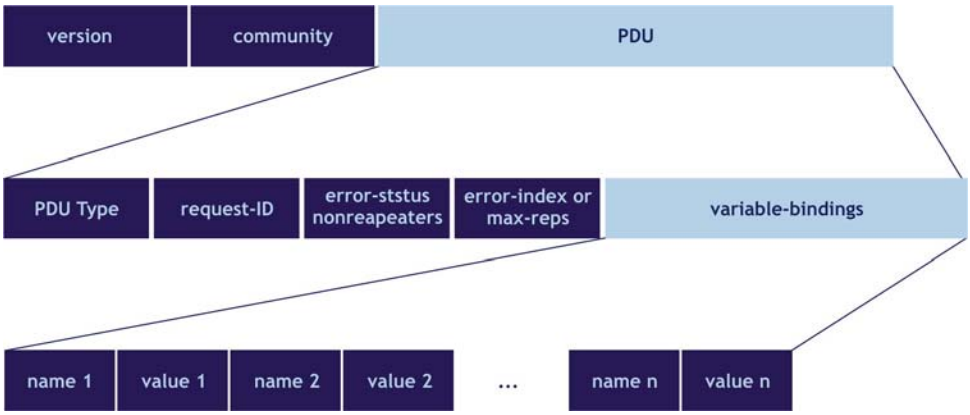


Figura 18 – Estrutura da mensagem SNMPv2

A mensagem SNMPv2, transportada sobre um datagrama UDP, contém a versão do protocolo, a *community* e o *Protocol Data Unit* (PDU) SNMP que contém a informação concreta sobre o tipo de pedido e a informação trocada. Esta figura permite-nos constatar claramente a falta de segurança inerente ao uso do protocolo. Nas versões 1 e 2, o mecanismo de segurança no acesso aos dados de gestão assenta no uso de *community strings*, passadas em claro na mensagem SNMP ou seja, sem qualquer tipo de encriptação. Este fraco mecanismo de segurança permite, através do uso de *sniffers* de rede, que as *community strings* possam facilmente ser lidas por terceiros.

3.6.3 Considerações Funcionais

Tal como próprio nome indica, o SNMP é um protocolo simples para a gestão, essencialmente, de equipamentos de rede como *routers* e *bridges*. Na altura em que o protocolo foi desenvolvido, os requisitos de gestão eram pouco complexos além de que a capacidade de processamento do *hardware*, na altura, era bastante limitada pelo que havia que desenvolver um protocolo simples e que não sobrecarregasse os equipamentos. Contudo, a evolução tecnológica ao nível de equipamentos e sistemas deixou para trás o SNMP. Os

sistemas tornaram-se bastante mais complexos e a sua capacidade de processamento aumentou exponencialmente dando a origem a novas aplicações com mais funcionalidades transformando a gestão numa tarefa bastante mais complexa. O SNMP, dada a sua natureza "simples" não acompanha actualmente as necessidades de gestão deste novo mundo tecnológico.

Tendo sido desenvolvido para gestão de redes e respectivos equipamentos, o SNMP não está preparado para lidar por exemplo com elementos de *software* ou gestão de sistemas pese embora a existência de algumas MIB's que permitem o acesso a variáveis simples como a capacidade de armazenamento de discos ou *software* instalado em computadores. Esta é precisamente uma das áreas onde esbarramos com as limitações do SNMP. Seria extremamente interessante e útil poder efectuar desinstalações de *software* remotamente quer por questões de segurança quer por políticas de controlo de utilização de recursos. Contudo, o SNMP não o permite.

Além desta "fraca" capacidade de actuação do gestor SNMP no equipamento gerido, a utilização do SNMP sofre ainda de uma outra desvantagem no que concerne ao seu uso na gestão de computadores. Enquanto que nos equipamentos de rede, o SNMP é um dado adquirido, ou seja, qualquer equipamento tem o SNMP activo, nos computadores pessoais e servidores, o serviço SNMP é muitas vezes desligado por questões de segurança o que impede qualquer tipo de comunicação com o sistema gestor.

4 Projecto e Implementação

A implementação traduz o trabalho colocado em prática e a utilização real das normas, *standards* e protocolos apresentados no capítulo anterior. Foi desenvolvida uma plataforma de gestão para um parque informático, baseada essencialmente em WMI, com funcionalidades quer de monitorização *online* quer de monitorização *offline*. O administrador poderá, a qualquer momento, ter uma visão global sobre o universo gerido e tomar eventuais medidas que considere necessárias. A monitorização *offline* permite que, além da obtenção de históricos das máquinas, o administrador possa, por exemplo, verificar a ocorrência de alterações ao nível do *hardware* das máquinas, detectando-as por visualização de relatórios ou através do serviço de alertas e notificações. Este serviço permite configurar a plataforma para que a ocorrência de eventos de interesse (como a substituição de um disco rígido) despolete um aviso a um ou mais utilizadores da plataforma de gestão registados para esse evento. Para uma maior clarificação do capítulo, os utilizadores da plataforma de gestão serão considerados técnicos e referenciados como tal nas alusões que lhes serão feitas no texto que se segue.

Este capítulo encontra-se dividido em quatro secções. Na secção **4.1** são apresentadas algumas das decisões que estiveram na base das escolhas tecnológicas para o desenvolvimento da plataforma. De seguida, em **4.2** é feita uma introdução à plataforma de gestão desenvolvida e à arquitectura da mesma. A terceira secção, **4.3**, contempla os detalhes de implementação ao nível do sistema de informação que suporta a plataforma. Por último, em **4.4**, é feita uma breve referência às ferramentas e tecnologias de programação usadas para o desenvolvimento da plataforma de gestão.

4.1 Plataforma de Monitorização/Gestão

No sentido de colocar em prática e demonstrar a utilidade do modelo de informação comum e das tecnologias desenvolvidas pelo DMTF para a gestão centralizada de máquinas, foi desenvolvida uma plataforma de monitorização e gestão centralizada de máquinas. Dadas as tendências actuais para a *webização*, foi decidido fazer o desenvolvimento da plataforma em um ambiente *web*, com a integração de várias ferramentas e tecnologias no sentido de disponibilizar uma interface única e universal ao administrador dos sistemas. Um dos

principais motivos para o desenvolvimento em ambiente *web* prende-se sobretudo com a universalidade do protocolo HTTP e com a facilidade da utilização de linguagens (como o PHP) quer para a construção da interface de apresentação quer para o sistema de *backend* onde foi desenvolvida toda a lógica e funcionalidades do sistema.

De acordo com a filosofia do DMTF, no desenvolvimento da plataforma de gestão houve uma tentativa de utilização de tecnologias abertas e universais no sentido de abrir o leque do universo de máquinas a gerir. Contudo, e dado que as iniciativas desenvolvidas pelo DMTF estão ainda numa fase inicial e com reduzida implementação nos sistemas operativos actuais, optou-se por estreitar o espaço de gestão reduzindo-o a máquinas com o sistema operativo Microsoft. A principal razão para esta escolha reside no facto de este tipo de sistema operativo suportar nativamente a iniciativa WBEM, a partir da qual é possível monitorizar e gerir as máquinas remotamente. A desvantagem óbvia desta escolha está na implementação proprietária da iniciativa no sistema operativo. Conforme foi já dito na subsecção **3.5**, a implementação WBEM da Microsoft recorre a objectos COM e DCOM [44] (tecnologia proprietária da Microsoft) pelo que a integração com outro tipo de sistemas ficou desde logo comprometida. Como resultado desta escolha, definiu-se logo à partida um requisito essencial para o desenvolvimento da plataforma de gestão: esta teria que ser desenvolvida sobre um sistema operativo da Microsoft para se poder utilizar os referidos objectos COM/DCOM.

A linguagem de programação a usar deveria também suportar a instanciação dos mesmos objectos. A escolha recaiu sobre a versão 5 do PHP. Relativamente ao sistema de informação, a escolha do mesmo é irrelevante desde que a base de dados seja relacional e suporte a interrogação SQL. A opção recaiu sobre o MySQL. Uma outra componente do sistema que será descrita em mais detalhe nas secções seguinte é a utilização de tarefas automáticas para a recolha de dados das máquinas. Para tal foi necessário integrar este módulo com o sistema operativo em uso, recorrendo-se a *scripts* em Perl dado que já existe um módulo desenvolvido para máquinas com o sistema operativo Windows que disponibiliza todas as funções necessárias para a criação de *schedule jobs*.

4.2 Arquitectura

O sistema de gestão centralizada desenvolvido é constituído por dois grandes blocos: o subsistema de apresentação e o subsistema de gestão conforme ilustrado na figura 19 na página seguinte.

O primeiro, como o próprio nome indica, serve fundamentalmente para apresentação de resultados e informação sobre o universo de máquinas gerido. Representa portanto a interface entre o administrador e o subsistema de gestão. Através desta interface, o administrador poderá consultar em tempo real, informações e configurações das máquinas

geridas bem como criar tarefas (automáticas) que lhe permitam registar em base de dados as informações ou configurações obtidas a partir das máquinas geridas. Existe também um módulo de alertas e notificações que permite ao administrador criar e configurar alertas e notificações para que possa ser notificado sobre eventuais situações anómalas ou de particular interesse (ex.: espaço em disco acima de 90% de ocupação).

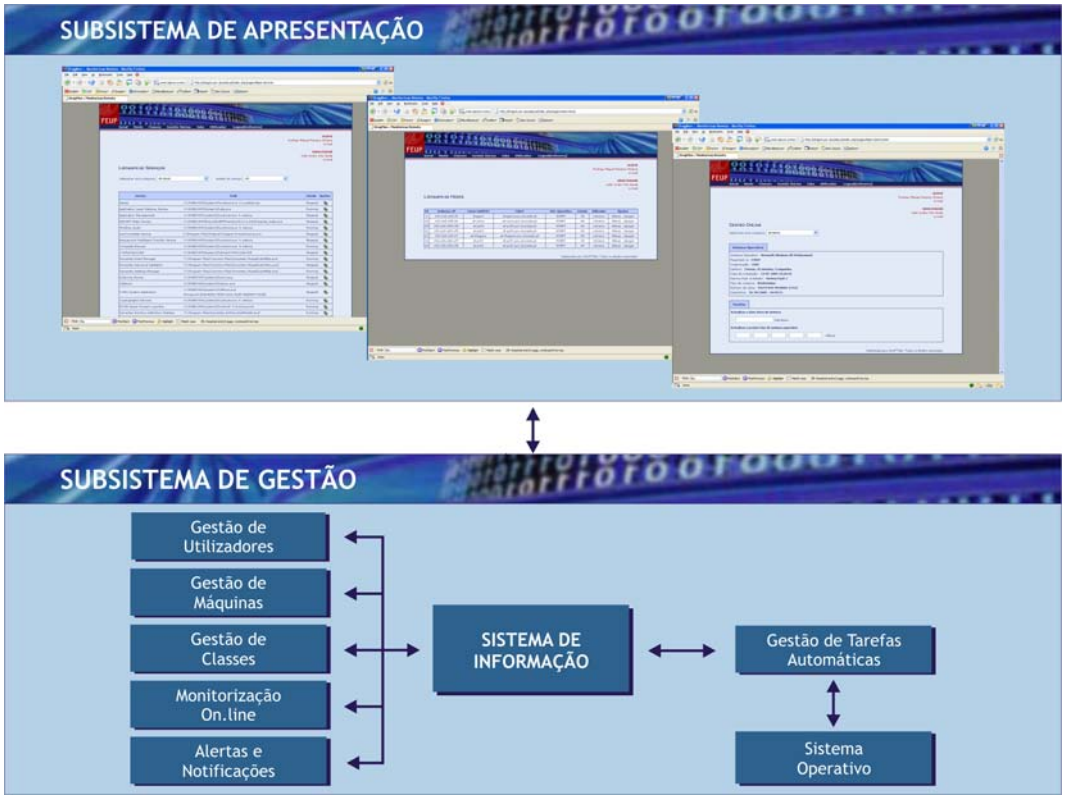


Figura 19 – Subsistemas da plataforma de gestão

No subsistema de gestão, reside toda a lógica e funcionalidades disponibilizadas pelo sistema global de gestão centralizada. Desenvolvido de forma modular, e facilmente expansível, é neste momento constituído por seis módulos, de acordo com o tipo de funcionalidades disponibilizadas. O elemento agregador de todos os módulos é um sistema de informação onde se encontra armazenada toda a informação relacionada quer com o subsistema de apresentação quer com o subsistema de gestão.

Os vários módulos que constituem o subsistema de gestão são:

- **Gestão de utilizadores** – responsável pela gestão de acessos, autorizações e privilégios de utilizadores da plataforma de gestão.
- **Gestão de máquinas** – permite adicionar ou remover máquinas do sistema de gestão. Permite ainda actualizações das características básicas de identificação das máquinas (nome, estado, endereço IP, etc.).

- **Gestão de classes** – Este módulo permite ao administrador adicionar ou remover classes CIM bem como propriedades e métodos das mesmas. Desta forma, é possível configurar os mecanismos de recolha de informação na fase de monitorização de acordo com as necessidades do administrador além de permitir que a qualquer momento sejam configurados novos processos de recolha de informações e configurações para classes novas.
- **Monitorização online** – Este módulo encontra-se dividido em várias subcategorias que permitem ao administrador monitorizar e obter em tempo real informações sobre uma máquina em particular. As subcategorias disponibilizadas são: sistema operativo, computador, *hardware*, memória, utilizadores, processos, serviços, configuração de rede, conexões de rede e *software* instalado.
- **Monitorização offline** – A plataforma de gestão permite ao administrador consultar o histórico de informações obtidas para o universo de máquinas geridas efectuando simples consultas ao sistema de informação.
- **Alertas e notificações** – O sistema de alertas e notificações permite ao administrador configurar determinadas condições ou conjunto de condições que no caso de ocorrerem em determinadas máquinas deverão despoletar notificações e/ou alertas.
- **Gestão de tarefas automáticas** – Para automatizar o processo de gestão, eliminando a necessidade constante de o administrador verificar o estado das máquinas e respectivas configurações, foi desenvolvido um módulo de gestão de tarefas automáticas que, integrado com as funcionalidades de tarefas automáticas do sistema operativo, permite que o administrador crie tarefas cujo objectivo seja a recolha de dados e configurações a partir de um conjunto de máquinas de acordo com critérios específicos e subsequente registo no sistema de informação.

Em termos de funcionamento, o subsistema de gestão pode ser subdividido em dois. Por um lado, a monitorização pode ser feita *online*, ou seja, o administrador pode obter informação ou actuar sobre as máquinas geridas em tempo real. Por outro lado, e em complemento à monitorização *online*, o administrador pode também configurar tarefas automáticas para recolha da informação de gestão das máquinas para posteriores consultas e elaboração de relatórios com o histórico das máquinas – monitorização *offline*. A figura 20 na página seguinte representa a arquitectura e modo de funcionamento de acordo com o tipo de monitorização escolhida pelo administrador.

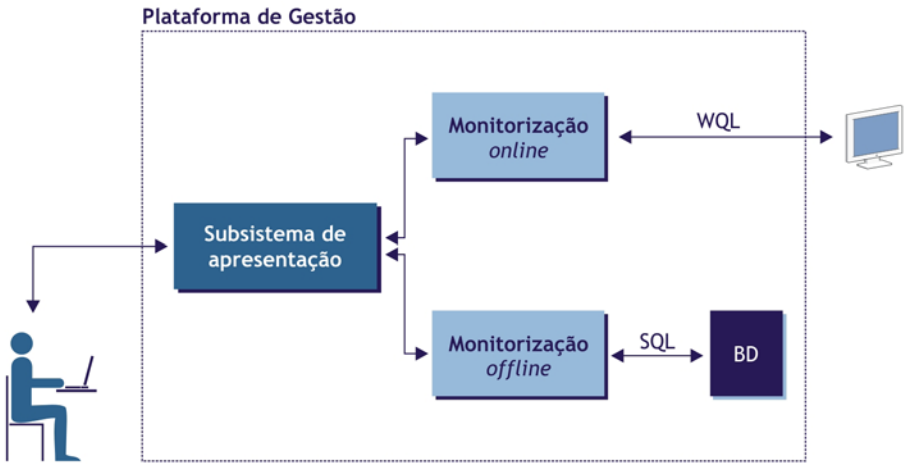


Figura 20 – Arquitectura da plataforma de gestão

Note-se que a monitorização online é feita usando *WMI Query Language* (WQL) [51] para obtenção da informação em tempo real enquanto que a monitorização *offline* recorre a puras consultas à base de dados usando a tradicional linguagem de acesso a bases de dados relacionais – *Structured Query* Language (SQL) [50].

4.3 Sistema de Informação

O sistema de informação, agindo como um elemento agregador de todos os módulos do subsistema de gestão, é responsável pelo armazenamento de toda e qualquer tipo de informação inerente ao funcionamento da plataforma. Aqui se inclui a informação de gestão recolhida a partir das máquinas, as características das máquinas pertencentes ao universo gerido, as tarefas automáticas configuradas pelo administrador, descrição das classes CIM, propriedades e métodos, informação relativa aos utilizadores do sistema de gestão, etc.

No seguimento da subdivisão do subsistema de gestão em módulos, a descrição do sistema de informação será feita de acordo com os mesmos para um melhor entendimento da sua estrutura. Serão descritos os subsistemas de informação relativos a:

- utilizadores e grupos de utilizadores da plataforma
- classes CIM, propriedades e métodos
- máquinas e grupos de máquinas
- tarefas automáticas
- alertas e notificações

4.3.1 Utilizadores e Grupos

Apesar de não estar directamente relacionada com a gestão remota de máquinas, a gestão de utilizadores é um aspecto essencial para garantir a segurança em todo o processo. Dado que o sistema lida com configurações de máquinas, tornou-se necessário introduzir aqui

um mecanismo de autenticação e autorização. A figura 21 representa a estrutura de dados desenvolvida.

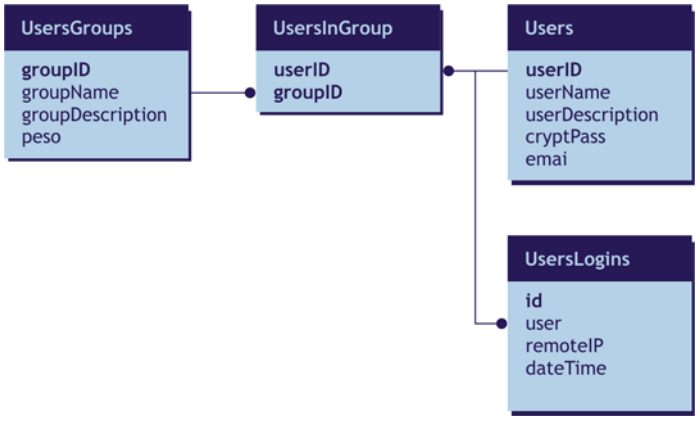


Figura 21 – Modelo de informação para autenticação e autorização

Os utilizadores são caracterizados por um *username*, uma descrição, um endereço de correio electrónico e a respectiva palavra-passe. Esta informação encontra-se na tabela **Users**. Existem também grupos de utilizadores (tabela **UsersGroups**) caracterizados por um nome, uma descrição e um peso. O peso permite criar níveis hierárquicos (autorização) no acesso à informação disponibilizada pela plataforma e à execução de determinadas tarefas sobre as máquinas geridas. Um utilizador pode pertencer a mais do que um grupo, bastando para tal a inserção do respectivo registo na tabela que relaciona as duas entidades (**UserInGroup**). Uma última funcionalidade foi ainda criada: o registo das autenticações dos utilizadores da plataforma. Com esta funcionalidade pretende-se apenas disponibilizar informação extra ao utilizador, aumentando o grau de segurança do sistema e oferecendo também um histórico de autenticações dos utilizadores da plataforma.

4.3.2 Classes CIM, Propriedades e Métodos

Para efeitos de identificação de classes e respectivos métodos e propriedades, foram criadas algumas tabelas que armazenam toda essa informação. Dada a imensidão de informação contida no modelo CIM e nos modelos de extensão, está armazenada apenas uma parcela do modelo, contendo as classes consideradas pertinentes para o efeito deste trabalho. No entanto, a plataforma de gestão desenvolvida permite ao utilizador adicionar ou remover, a qualquer instante, classes, propriedades ou métodos.

A criação de uma estrutura de armazenamento da informação relativa às classes permite que a posterior recolha de dados das estações remotas monitorizadas/geridas seja totalmente flexível e adaptável às necessidades do utilizador/administrador. Para a criação desta estrutura houve que ter em conta o seguinte:

Tabela 2 – Pré-requisitos para a estrutura de dados de classes

1	Classes, propriedades e métodos são caracterizados por um nome e uma descrição
2	Uma classe pode ter várias propriedades e vários métodos
3	Uma propriedade ou um método pertencem a apenas uma classe
4	Podem existir vários métodos ou propriedades com o mesmo nome, desde que pertencentes a classes distintas
5	Uma propriedade tem um determinado tipo de dados
6	Um método pode ter mais do que um parâmetro
7	Os parâmetros de um método têm um determinado tipo de dados

Tendo em conta estes requisitos, foi desenvolvida a seguinte estrutura (figura 22).

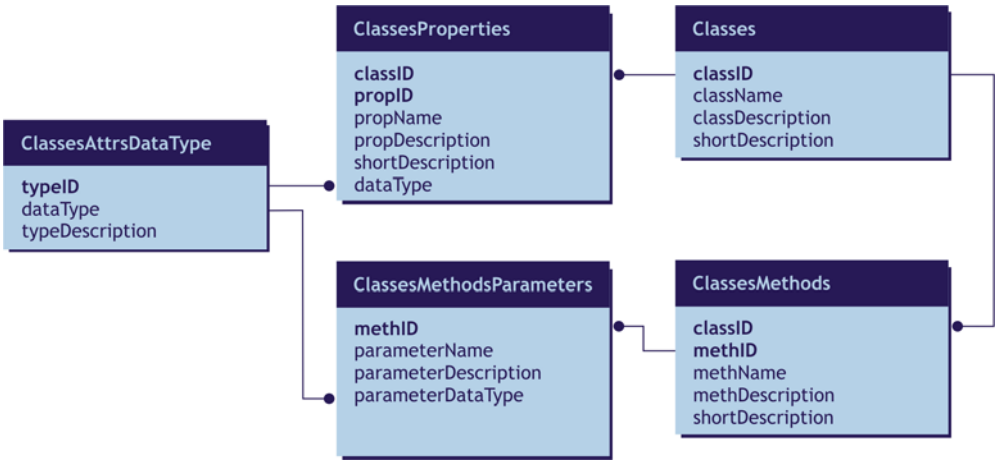


Figura 22 – Modelo de informação para classes, propriedades e métodos CIM

As tabelas que constituem a estrutura apresentada são:

- **Classes**: classes do modelo de informação comum e de extensões ao modelo. As classes são caracterizadas por um nome e uma descrição e identificadas por um identificador único: **classID**.
- **ClassesProperties**: propriedades das classes constantes na tabela **Classes**. As propriedades são caracterizadas por um nome, uma descrição e um tipo de dados e são identificadas pelo **classID** da classe a que pertencem e um identificador próprio, único dentro da respectiva classe: **propID**.
- **ClasseMethods**: métodos das classes constantes na tabela **Classes**. Os métodos são caracterizados por um nome e uma descrição e identificados pelo **classID** da classe a que pertencem e um identificador próprio, único dentro da respectiva classe: **methID**.

- **ClassesMethodsParameters**: parâmetros dos métodos constantes na tabela **ClassesMethods**, no caso de existirem alguns. Os parâmetros são caracterizados pelos identificadores da classe e do método a que pertencem, por um nome, uma descrição e um tipo de dados e são identificados pelo campo **ID**.
- **ClassesAttrsDataType**: tipos de dados possíveis para propriedades e métodos de classes. São caracterizados por um tipo de dados e uma descrição.

4.3.3 Máquinas e Grupos de Máquinas

O armazenamento de informação relativa às máquinas do ambiente gerido contempla duas áreas. Por um lado temos a informação genérica acerca das mesmas, onde se inclui aspectos como a identificação da máquina, o sistema operativo, o estado e eventuais grupos a que as máquinas possam pertencer. Por outro lado, há que armazenar a informação específica acerca do *hardware* e dos componentes que compõem a máquina. Esta informação disponibiliza ao administrador do parque informático o conhecimento, em qualquer altura, da composição de uma máquina em termos físicos e, consequentemente, de eventuais alterações realizadas nas máquinas.

Para o armazenamento da informação genérica das máquinas geridas foi desenvolvida uma estrutura simples, com a capacidade de as organizar em grupos. Esta característica facilita posteriormente a gestão pois permite ao administrador executar determinadas tarefas por grupo em vez de individualmente para além de poder criar alertas e/ou notificações específicas para um conjunto particular de máquinas.

Para tal, foram considerados os seguintes requisitos (tabela 3):

Tabela 3 – Pré-requisitos para o modelo de informação genérica de máquinas

1	As máquinas são caracterizadas por um conjunto de características: nome, endereço IP, sistema operativo, estado, etc.
2	Existem vários tipos de sistemas operativos
3	Existem vários estados possíveis
4	Uma máquina pode pertencer a mais do que um grupo de máquinas
5	Um grupo de máquinas é caracterizado por um nome e uma descrição

Na página seguinte, a figura 23 ilustra a estrutura desenvolvida para o efeito.

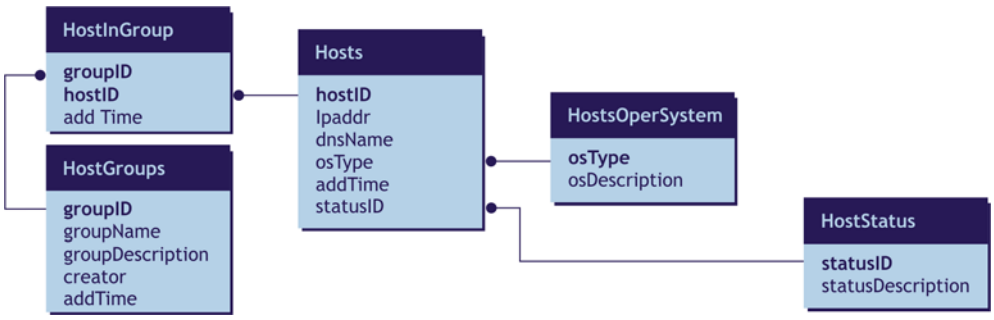


Figura 23 – Modelo de informação para máquinas e grupos de máquinas

De acordo com a figura apresentada, as tabelas que constituem este subsistema de informação são:

- **Hosts**: contém informação acerca de todas as máquinas do universo de gestão. Uma máquina é identificada pelo **hostID**.
- **HostsOperSystem**: esta tabela contém as descrições dos sistemas operativos que poderão estar instalados nas máquinas a gerir.
- **HostsStatus**: esta tabela contém os vários estados possíveis para as máquinas pertencentes ao universo de gestão (ON, OFF, TESTING).
- **HostGroups**: aqui encontram-se todos os grupos criados pelo(s) administrador(es). Os grupos de máquinas são identificados por um **groupID** e caracterizados por um nome e uma descrição.
- **HostInGroup**: imputação das máquinas a grupos.

Quanto à informação específica relativa ao *hardware* das máquinas, o sistema de informação é um pouco mais complexo dado o leque alargado de dados a armazenar. Assim, e ao nível da plataforma de gestão desenvolvida, considerou-se que uma máquina é composta pelos seguintes componentes:

- processador
- *motherboard*
- BIOS
- leitor CD/DVD
- memória
- placa gráfica
- adaptador de rede
- placa de som
- disco rígido
- teclado
- rato

Na realidade, uma máquina pode ser composta por bastantes mais componentes do que os apresentados. Contudo, para efeitos da plataforma desenvolvida, consideraram-se estes

como os mais importantes e aqueles que, do ponto de vista da gestão de um parque informático, requerem uma monitorização mais atenta. Note-se também que, qualquer máquina pode conter em si múltiplos componentes do mesmo tipo. Por exemplo, são extremamente vulgares máquinas com mais do que um disco rígido e vários leitores de CD/DVD.

Um outro requisito que foi considerado na fase de projecto do sistema foi a capacidade de monitorizar os componentes, mantendo um histórico dos mesmos. Por exemplo, se um componente como uma placa gráfica ou uma placa de som aparece numa outra máquina, dever-se-á manter o histórico do componente de forma a identificar o seu "percurso" no parque informático gerido. Para tal é necessário considerar os componentes individualmente como unidades independentes. Ao nível do sistema de informação foi necessário criar estruturas de dados capazes de suportar este requisito ou seja, para cada tipo de componente, foi criada uma tabela com propriedades específicas acerca desse componente e, através do relacionamento dessas tabelas com a tabela de máquinas, identificar que componentes constituem uma máquina.

Na figura 24 é apresentada parte da estrutura de dados desenvolvida para o efeito, sendo que a tabela **Hosts** (com informação genérica relativa às máquinas) é a mesma da figura 23. Por questões de simplicidade, as restantes tabelas que compõem a figura 23 foram retiradas.

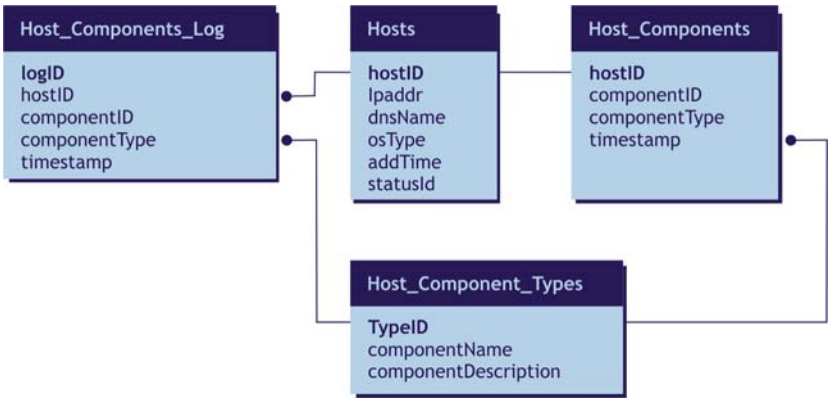


Figura 24 – Constituição de um *host* em componentes

Uma máquina, representada por um registo da tabela **Hosts** é constituída por vários componentes, cada um deles identificado por um código (**componentID**) e de um determinado tipo, descrito na tabela **Host_Component_Types**. A tabela **Host_Components_Log** permite guardar o histórico dos componentes nas máquinas.

Os vários componentes que constituem uma máquina são descritos, de acordo com o seu tipo, pela correspondente tabela na base de dados. A título de exemplo, a figura 25 apresenta a estrutura das tabelas de informação relativas aos componentes *motherboard*, placa de som e carta de rede.

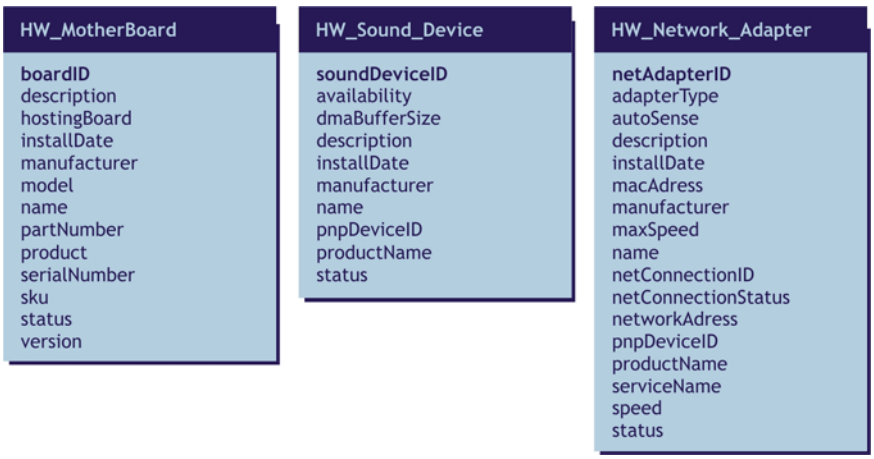


Figura 25 – Informação de componentes de uma máquina

Note-se que cada componente é descrito por um conjunto de propriedades específicas, herdadas das classes CIM, que os representam. Por exemplo, a descrição do componente *motherboard* pode ser obtida por consulta à classe `Win32_BaseBoard`. Em cada uma das tabelas, a identificação do componente é feita através do respectivo ID (**boardID**, **soundDeviceID** e **netAdapterID**) que se relacionará com o atributo **componentID** das tabelas **Host_Components** e **Host_Components_Log**.

4.3.4 Tarefas Automáticas

O módulo de gestão de tarefas é o único que requer a integração com o sistema operativo da máquina onde se encontra instalado o sistema de gestão. Como tal, foi necessário desenvolver uma estrutura que servisse de suporte quer à plataforma de apresentação para a listagem, edição e criação de tarefas automáticas quer ao subsistema de gestão, nomeadamente ao módulo de gestão de tarefas automáticas e consequente integração com o sistema operativo. Assim, as estruturas de dados foram criadas tendo por base as características das tarefas automáticas disponibilizadas pelo sistema operativo.

Antes de iniciarmos a descrição mais detalhada do modelo e respectivas tabelas, importa referir alguns dos requisitos iniciais que foram considerados bem como as necessidades implícitas a um sistema de gestão de tarefas automáticas tendo em conta, obviamente, a integração com o sistema operativo. Nesse sentido, foram identificadas inicialmente o tipo de tarefas que poderiam ser criadas pelo administrador em termos da respectiva periodicidade. Dado que o sistema de gestão foi desenvolvido sobre a plataforma *Windows*, o levantamento realizado permitiu considerar as seguintes periodicidades para as tarefas automáticas:

– **Executar apenas uma vez**, sendo a tarefa caracterizada por uma data e hora de execução.

– **Executar com periodicidade horária**, sendo a tarefa caracterizada por uma data e hora de execução bem como um intervalo de repetição e os dias da semana em que será executada.

– **Executar com periodicidade diária**, sendo a tarefa caracterizada por uma hora de execução e os dias em que será executada.

– **Executar com periodicidade semanal**, sendo a tarefa caracterizada por uma hora e dia de execução.

– **Executar com periodicidade mensal**, sendo a tarefa caracterizada por uma hora de execução bem como o dia do mês e meses em que será executada.

Tendo como base de trabalho os referidos requisitos e as diferentes periodicidades, foi desenvolvido o seguinte modelo de dados (figura 26).

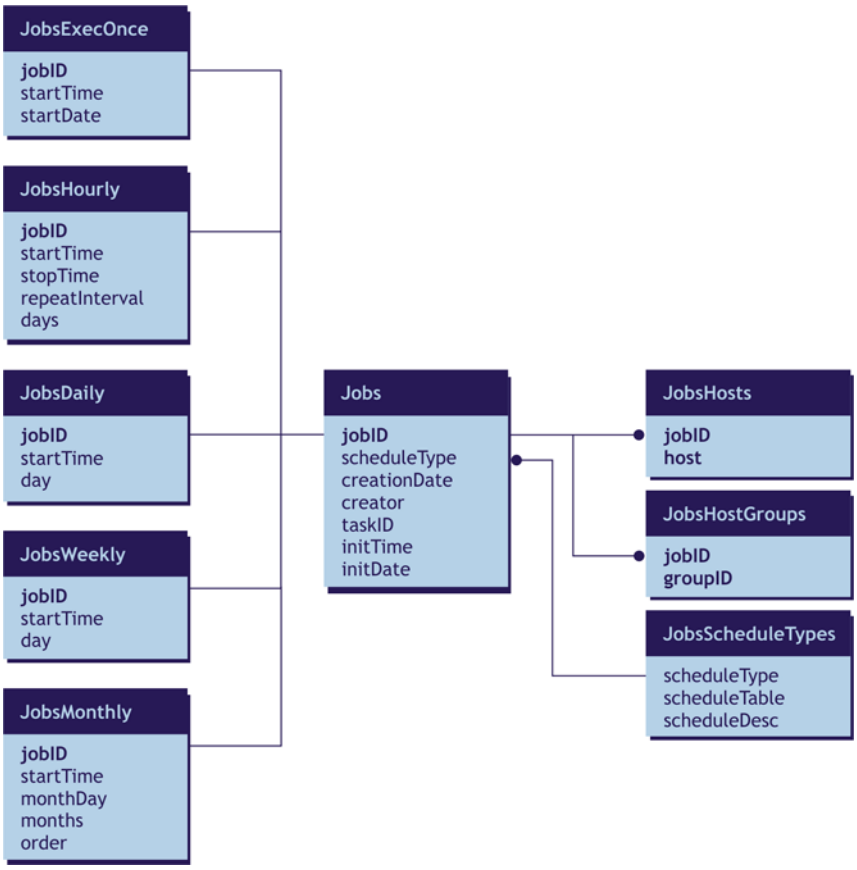


Figura 26 – Modelo de informação para tarefas automáticas

Tendo em conta as periodicidades configuráveis para as tarefas automáticas, foram criadas as respectivas tabelas para armazenamento das configurações das tarefas. Segue-se a descrição das tabelas constituintes.

- **JobsExecOnce**: configurações das tarefas com execução de apenas uma vez (data/hora de execução).
- **JobsHourly**: configurações das tarefas com periodicidade horária (data/hora, intervalo de repetição e dias de execução)

- **JobsDaily**: configurações das tarefas com periodicidade diária (hora e dias de execução)
- **JobsWeekly**: configurações das tarefas com periodicidade semanal (hora e dia de execução)
- **JobsMonthly**: configurações das tarefas com periodicidade mensal (hora, dia ou dias do mês e meses de execução)
- **Jobs**: contém informação genérica sobre todas as tarefas automáticas criadas. Qualquer uma das tarefas descritas é sempre caracterizada por uma data de criação, o utilizador que as criou, uma data e hora a partir da qual a tarefa poderá ser executada e um identificador (**taskID**) do tipo de acção que será executada sobre a(s) máquina(s) alvo.
- **JobsScheduleTypes**: tipos de tarefas, classificadas por periodicidade. Permite saber para cada tarefa criada, qual o seu tipo e, consequentemente, em que tabela se poderá encontrar informação detalhada sobre a configuração da execução da tarefa.
- **JobsHosts**: no caso de uma tarefa ser aplicada a uma máquina, esta tabela identifica o alvo.
- **JobsHostGroups**: tem o mesmo objectivo da tabela anterior só que aplicado a grupos de máquinas.

4.3.5 Alertas e Notificações

Em primeiro lugar, é necessário introduzir o conceito de alerta e o de notificação. Para efeitos do trabalho desenvolvido, foi considerado que um alerta é derivado de um acontecimento ou evento de elevado interesse para o(s) administrador(es) enquanto que uma notificação representa um acontecimento ou evento de baixo interesse. Quer para os alertas quer para as notificações, poderá haver mais do que um receptor.

Assim, o envio de um alerta requer que o receptor tome conhecimento do mesmo ou seja, o alerta será considerado “entregue” assim que o receptor desse alerta verifique, na plataforma de gestão, os acontecimentos que o despoletaram. Caso o receptor do alerta não o faça, continuará a ser alertado, eventualmente, com uma frequência maior. Eventualmente, os alertas podem ter algum grau de escalonamento, sendo enviados para utilizadores da plataforma de níveis hierárquicos superiores. A notificação, por sua vez, é entregue apenas uma vez ao receptor, cabendo-lhe depois indagar das razões da notificação ou simplesmente ignorá-la. Esta é, do ponto de vista da implementação, a única diferença entre um alerta e uma notificação. Na fase de criação e configuração quer dos alertas, quer das notificações, a interface apresentada ao administrador é exactamente a mesma e as operações permitidas são iguais para ambas.

Antes de avançar com a apresentação da estrutura de dados, importa salientar alguns aspectos que foram tomados em consideração durante a fase de projecto (tabela 4):

Tabela 4 – Pré-requisitos para o sistema de alertas e notificações

1	A diferença entre um alerta e uma notificação reside no grau de severidade do mesmo (quantificação do interesse).
2	O alerta e/ou notificação poderá ser aplicado a uma máquina ou a um conjunto de máquinas.
3	O envio do alerta e/ou notificação poderá ser feito para um administrador ou para vários administradores.
4	Existem vários tipos de alertas e notificações, podendo ser classificados por categoria.
5	Poderá haver a necessidade de combinar condições para o despoletar de um alerta e/ou notificação.
6	A periodicidade com que o sistema de gestão deverá verificar o estado das máquinas configuradas para um alerta deverá ser indicada na fase de criação do mesmo.
7	O sistema deverá ser o mais flexível possível, de modo a permitir ao administrador a criação de vários tipos de alertas e notificações.

A figura 27 representa a estrutura de dados desenvolvida.

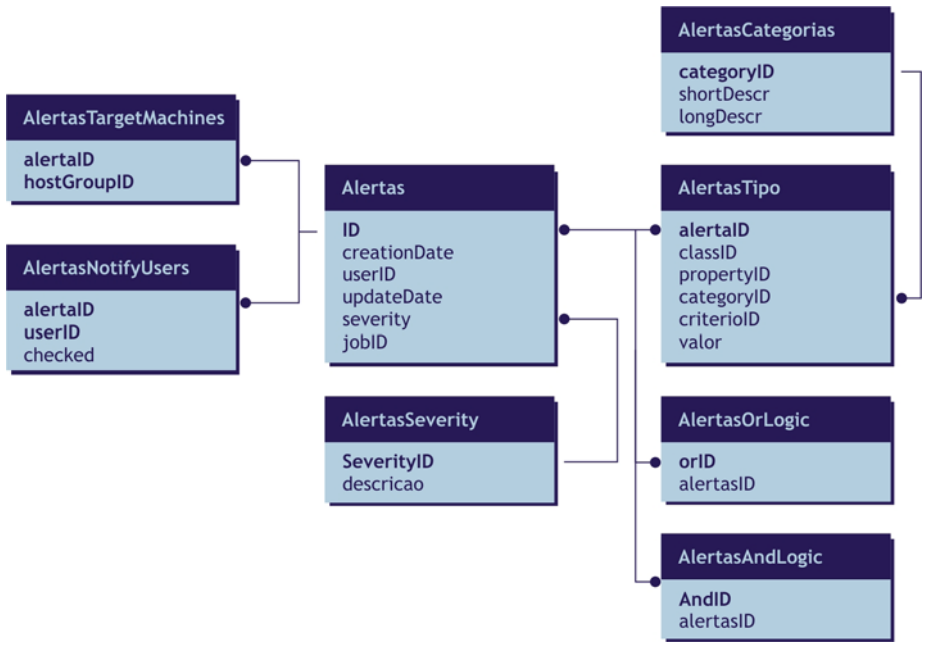


Figura 27 – Modelo de informação para o sistema de alertas e notificações

Um alerta/notificação é representado por um registo da tabela **Alertas** que contém a informação básica relativa ao mesmo como a data de criação, o utilizador que o criou, o grau de severidade (Tabela 4, ponto 1) e a data da última actualização do alerta/notificação. Note-se ainda o campo *jobId*, ao qual corresponderá uma tarefa de execução automática, de acordo com a configuração de periodicidade do alerta e/ou notificação (Tabela 4, ponto 6).

No seguimento do ponto 7 (Tabela 4), o modelo desenvolvido permite criar e configurar essencialmente dois tipos de alertas e/ou notificações: pré-definidos (divididas por categorias) e avançados.

Relativamente aos alertas pré-definidos, existe um conjunto de categorias de gestão (à semelhança da monitorização *online* que será vista na secção seguinte) como por exemplo o sistema operativo, o *hardware*, a configuração de rede, etc (**AlertasCategorias**) e em cada uma dessas categorias existe um subconjunto de tipos de alertas que o administrador pode escolher (Tabela 4, ponto 4). Quanto à configuração avançada de alertas e notificações, foi feita uma integração com o modelo de informação desenvolvido para as classes e propriedades CIM (**AlertasTipo**), podendo o administrador escolher directamente que propriedade de que classe será utilizada para a criação do alerta e/ou notificação. A combinação de condições no mesmo alerta e/ou notificação poderá ser representada quer em **AlertasAndLogic** quer em **AlertasOrLogic** (Tabela 4, ponto 5). Neste caso, a conjugação de condições será feita de acordo com a lógica pretendida: `AND` e `OR`.

A aplicação do alerta e/ou notificação a uma ou mais máquinas (Tabela 4, ponto 2) e o envio para um ou mais utilizadores (Tabela 4, ponto 3) é feito através de **AlertasTargetMachines** e **AlertasNotifyUsers** respectivamente.

4.4 Ferramentas e Tecnologias

Para o desenvolvimento da plataforma de gestão e das suas funcionalidades, apresentadas nas secções anteriores deste capítulo, foi necessário recorrer a um conjunto de ferramentas e tecnologias, tendo sempre por base o modelo de informação comum CIM, que se constituiu como o elemento central no desenvolvimento do trabalho realizado. Isto porque, mesmo variando o modo como a informação de gestão é apresentada ou alterando os métodos de acesso à mesma, a sua organização e estrutura mantém-se inalterável, conferindo à plataforma uma base consistente e universal, respeitando os *standards* definidos para o efeito.

Assim, o desenvolvimento da plataforma iniciou-se pela escolha de um sistema de armazenamento de informação central, de modo a permitir uma gestão *offline*, além de providenciar o suporte da plataforma de apresentação em termos das suas funcionalidades básicas como por exemplo um sistema de autenticação e autorização. De seguida, foram escolhidas as linguagens de programação a utilizar e o *framework* de apresentação.

4.4.1 Armazenamento de Informação

Relativamente ao armazenamento de informação existe uma variedade de sistemas de armazenamento desde sistemas de bases de dados (relacionais, XML, orientadas a objectos,

etc.) até sistemas de informação baseados em estruturas de directório como é exemplo o *Lightweight Directory Access Protocol* (LDAP) [22][23].

No âmbito do trabalho desenvolvido foi escolhido para sistema de armazenamento de informação um sistema de base de dados relacional, o MySQL [52]. Contudo, e ainda numa fase de levantamento de tecnologias e estudo do estado da arte, foi colocada a hipótese de o *backend* de dados assentar sobre um serviço de directório como por exemplo o *OpenLDAP* [53]. Esta hipótese acabou por ser descartada dado que os sistemas de directório, apesar de extremamente rápidos na leitura de dados, são algo pesados na escrita dos mesmos e, dado que haveria a necessidade de escrita constante, optou-se pela solução de um sistema de base de dados relacional.

O MySQL é, sem dúvida, o sistema de gestão de base de dados *Open-Source* mais popular, desde há alguns anos. Actualmente, o sistema é desenvolvido, distribuído e suportado pela empresa MySQL AB que apesar da orientação comercial que impôs ao projecto, continua a distribui-lo gratuitamente. Trata-se de um sistema de gestão de bases de dados relacionais onde os dados são armazenados em diferentes tabelas o que lhe confere maior velocidade e flexibilidade. O SQL (*Structured Query Language*) é a linguagem normalizada usada para aceder a este tipo de bases de dados e está definida no ANSI/ISO SQL Standard [50]. O sistema funciona segundo o modelo cliente/servidor podendo residir na mesma máquina ou em máquinas distintas e suporta um vasto leque de diferentes *backends* de administração bem como diferentes aplicações clientes, ferramentas administrativas de gestão, API's, etc.

Em termos de mecanismos internos e modo de funcionamento como sistema de base de dados, a escolha do MySQL deveu-se sobretudo ao facto de se tratar de um projecto aberto, actual e com as capacidades e funcionalidades necessárias para o decorrer do trabalho realizado. Quer isto dizer que qualquer outro tipo de sistema de base de dados teria servido já que o essencial e o tema em que incide este trabalho pertencem a outra área.

4.4.2 Linguagens de Programação

Dado que a escolha inicial sobre o tipo de plataforma a desenvolver recaiu sobre o ambiente *web*, o PHP surgiu como a linguagem natural para o seu desenvolvimento. Trata-se de uma linguagem *open-source* especificamente concebida para ambientes *web* e com um manancial de potencialidades para o efeito. Para a elaboração de *scripts*, mais concretamente na integração da plataforma com o sistema operativo, principalmente para a criação e gestão de tarefas automáticas, o Perl apresentou-se como a linguagem mais adaptada para o efeito dado o seu potencial em termos de *scripting* e processamento e identificação de padrões e também graças ao desenvolvimento de módulos que disponibilizam directamente ao programador funções exclusivas para ambientes *Windows*.

4.4.2.1 PHP

PHP é uma linguagem que, na sua vertente mais divulgada, permite criar sites *web* dinâmicos, possibilitando uma interacção com o utilizador através de formulários, parâmetros do URL e *links*. Trata-se de uma linguagem desenvolvida para utilização na Internet e que permite a criação de páginas em tempo real. Por exemplo, num motor de busca, seria completamente inviável manter uma ou mais páginas para cada consulta efectuada. Na realidade, o que existe é um mecanismo de resposta que é capaz de, em tempo real, consultar dados e construir a(s) página(s) enviada(s) para o cliente.

Tipicamente, existem três áreas onde os *scripts* PHP podem ser usados:

- *Server-Side Scripting* – esta é a forma tradicional de uso da linguagem e para a qual foi desenvolvida. Para tal é necessário um *parser* de PHP (tipicamente um módulo do servidor Web), um servidor Web e um cliente Web (*browser*).
- *Command-Line Scripting* – um *script* PHP pode ser usada para executar determinadas operações em linha de comandos. Neste caso não são necessários nem o servidor Web nem um cliente Web. Tipicamente, *scripts* deste género são usados no *cron* em sistemas *nix para executar determinadas operações periódicas.
- *Desktop Applications* – apesar de nesta área existirem bastantes alternativas e, se calhar, com maior sucesso e simplicidade no desenvolvimento, existem algumas funcionalidades do PHP que permitem desenvolver ambientes gráficos.

Para o desenvolvimento da plataforma de gestão o PHP foi usado no seu modo tradicional ou seja, *server-side scripting*.

Actualmente existem distribuições de PHP para uma variedade de plataformas desde o *Linux* ao *Windows* passando pelo *MacOS*, *Solaris*, *HP-UX*, etc. Além disso, o PHP é suportado por uma variedade de servidores *web*: *Apache*, *Microsoft IIS*, *Personal Web Server*, etc. Esta acaba por ser também outra grande vantagem da linguagem. Ela pode ser usada praticamente em qualquer sistema operativo e com um vasto leque de servidores *web* com os quais pode ser integrado.

O lançamento da versão 5 veio alterar substancialmente o modelo de programação orientada a objectos introduzindo novas funcionalidades e corrigindo outras (é o caso da passagem de objectos para funções ser feita por referência em vez de cópia como acontecia na versão 4). Outra grande vantagem do PHP é a sua capacidade para suportar múltiplos sistemas de base de dados: MySQL, MS-SQL, PostgreSQL, Informix, ODBC, Ingres, etc.

Em resumo, o PHP apresentou-se como uma linguagem de programação extremamente versátil, *open-source*, e já com créditos firmados no que respeita ao

desenvolvimento em ambientes *web*. Aliada a estas capacidades natas da linguagem, há que acrescentar o suporte de objectos COM, condição essencial no desenvolvimento da plataforma dado que o seu funcionamento iria recorrer ao WMI.

No que concerne ao trabalho desenvolvido, o PHP foi usado para a construção do subsistema de apresentação e para o módulo de monitorização *online*, incluindo o acesso às máquinas, a recolha e a formatação dos dados de gestão a apresentar. Quanto à monitorização *offline*, sendo constituída por dois grandes blocos (apresentação dos dados recolhidos *offline* e tarefas automáticas para a recolha dos mesmos), foi necessário utilizar o Perl além do PHP. Isto porque um dos blocos contém toda a lógica de apresentação de informação ao administrador o que inclui parte do subsistema de apresentação. Por outro lado, a monitorização *offline* requer a criação de tarefas automáticas para a recolha remota da informação pelo que neste aspecto recorreu-se a um misto de *scripts* em Perl e PHP.

Do ponto de vista do WMI, o desenvolvimento de aplicações para a gestão de máquinas inclui normalmente 3 passos: a ligação remota à máquina, a elaboração e execução de uma *query* WQL e a obtenção e processamento dos dados retornados pela *query*. O código da figura 28 exemplifica a ligação a uma máquina remota.

```
$locator = new COM("WbemScripting.SWbemLocator");
$wbemServ = $locator->ConnectServer($host,$namespace);
```

Figura 28 – Código PHP para uma conexão remota via WMI

O primeiro passo para uma conexão remota passa pela obtenção de um apontador *IWbemLocator*, feito através da criação de um objecto COM *SWbemScripting.SWbemLocator*. O objecto *locator* permite efectuar posteriormente uma conexão ao serviço WMI de uma máquina através do método *ConnectServer*. Este método permite especificar uma máquina e um *namespace* ou, por omissão, cria uma ligação à máquina corrente no *default namespace*. O resultado da invocação deste método é um objecto *SWbemServices*, responsável depois pela comunicação com o CIMOM. Note-se que apenas se pode aceder a objectos presentes no espaço de nomes indicado na fase de ligação. Para obter objectos de *namespaces* diferentes seria necessário criar um outro objecto do tipo *SWbemServices* com outra chamada ao método *ConnectServer*.

Depois de criado um canal de comunicação entre a aplicação e o CIMOM na máquina remota, é colocada a *query* para a obtenção da informação desejada conforme a figura 29 na página seguinte.

```
/* Todas as instâncias da classe Win32_LogicalDisk */
$query_1 = "SELECT * FROM Win32_LogicalDisk";

/* Todas as instâncias da classe Win32_Service */
/* com o valor Stoped na propriedade State     */
$query_2 = "SELECT * FROM Win32_Service WHERE State='Stoped'";

/* Apenas as propriedades Name e FreeSpace de todas */
/* as instâncias da classe Win32_LogicalDisk        */
$query_3 = "SELECT Name,FreeSpace FROM Win32_LogicalDisk";

/* Execução da query */
$objects = $wbemServ->execquery($query_1);
```

Figura 29 – Código PHP para execução de uma *query* WQL

A *query* é executada através da invocação do método *execquery* sobre o objecto *SWbemServices*, obtido no passo anterior. A figura 29 apresenta alguns exemplos de *queries* e invocação do respectivo método.

Por último, depois de estabelecida a ligação e executada a *query* na máquina remota, resta processar a resposta. Na figura 30 encontra-se o extracto de código necessário para o processamento dos dados provenientes da execução de uma *query* do tipo `query_1` apresentada na figura 29.

```
foreach($objects as $object) {
   print $object->caption;       // c:
   print $object->description;   // Local fixed disk
   print $object->filsystem;     // NTFS
   print $object->size;          // 122926776320 (~120GB)
   print $object->freespace;     // 9241149440 (~9.2GB)
}
```

Figura 30 – Código PHP para processamento do resultado de uma *query* WQL

A resposta consiste numa lista de objectos em que cada objecto corresponde a uma instância da classe sobre a qual foi feita a *query* e os seus campos são as propriedades da classe. No extracto de código apresentado na figura 30 é feito o *print* de algumas das características de um disco (campos do objecto). Em forma de comentário, cada linha contém o resultado de uma *query* real.

Além deste método de ligação, o WMI disponibiliza ainda um outro método, pouco utilizado mas cuja simplicidade pode por vezes facilitar o acesso, dependendo do grau de complexidade das operações que se pretendem realizar. Trata-se dos *monikers* [54]. Os *monikers* são objectos COM que resumem num único passo a fase de ligação e de execução da *query* na máquina remota. O uso dos *monikers* consiste na obtenção de um objecto

SWbemServices que agrega num único passo a criação de um *SWbemLocator*, a invocação do método *ConnectServer* e a invocação do método de execução da *query*, *execquery*.

Apesar de simplificar o acesso WMI a uma máquina remota, o uso de *monikers* tem algumas desvantagens dadas as suas limitações. Por exemplo, não é possível alterar as credenciais de acesso ou seja, são usadas as credenciais do utilizador corrente o que pode limitar o acesso remoto. Por outro lado, dado que é feita uma agregação de operações num único passo, o método pode-se tornar ineficiente pois não há reutilização do objecto da ligação.

Por exemplo, a consulta anterior de dados da classe `Win32_LogicalDisk` poderia ser feita conforme a figura 31. Note-se que a identificação da máquina alvo é feita na instanciação do objecto COM, sendo necessário construir um *object path* completo: `//dragom.fe.up.pt/root/cimV2/Win32_LogicalDisk`.

```
$objects = new COM("WinMgmts://dragom.fe.up.pt/root/cimV2/Win32_LogicalDisk");
foreach($objects as $object) {
  print $object->caption;          // c:
  print $object->description;      // Local fixed disk
  print $object->filsystem;        // NTFS
  print $object->size;             // 122926776320 (~120GB)
  print $object->freespace;        // 9241149440 (~9.2GB)
}
```

Figura 31 – Utilização de *monikers* no acesso WMI

A grande vantagem do uso de *monikers* reside sobretudo na sua simplicidade à custa de um aumento significativo na dificuldade de construção dos mesmos (*object path*) para consultas mais complexas.

No Anexo B são apresentados exemplos mais completos, desenvolvidos para a plataforma de gestão, sobre o acesso remoto via WMI e a obtenção de informação de gestão sobre áreas específicas.

4.4.2.2 Perl

Perl é uma linguagem de programação desenvolvida inicialmente por Larry Wall e suportada actualmente por uma vasta comunidade aberta de desenvolvimento. Perl é o acrónimo de "*Practical Extraction and Report Language*". Trata-se de uma linguagem de *scripting* optimizada para manipulação de *strings*, operações de I/O e tarefas de sistema [55]. Existem extensões que cobrem praticamente todas as áreas de gestão de sistemas, particularmente para Unix o que torna a linguagem bastante popular entre administradores de sistemas.

Estas particularidades e capacidades da linguagem fazem dela uma das preferidas também para o desenvolvimento de aplicações de rede. A linguagem foi construída de raiz com o objectivo de facilitar a comunicação entre processos, vulgarmente designada por IPC – *InterProcess Comunication*. Outra das grandes capacidades da linguagem, e que a distinguiu de muitas outras, é o seu elevado grau de funcionalidade em termos de expressões regulares.

Tratando-se de um projecto *Open-Source*, e dado que a linguagem se organiza em diversos módulos, é possível encontrar todo o tipo de módulos para toda e qualquer tipo de funcionalidade. No site `http://www.cpan.org` pode ser encontrada uma vasta colecção de *software* e documentação. A variedade de *software* é imensa e podemos encontrar módulos tão distintos como por exemplo:

- WebService::GoogleMaps – interface para os mapas do Google.
- Spreadsheet::ParseExcel – API para leitura de folhas de cálculo em formato xls.
- Nmap::Scanner – API para execução de *scans* com o NMAP.
- Astro::Cosmology – Cálculo de tempos, distâncias e volumes cosmológicos.

Dadas as características do WMI e do modo como foi implementado o acesso remoto ao mesmo (objectos COM), em Perl, as consultas e acessos remotos são feitas de forma semelhante ao PHP. Uma consulta WMI a uma máquina remota envolve três passos: o estabelecimento da ligação, envio da *query* e a recepção e processamento dos dados. O código da figura 32 representa a referida consulta para a obtenção dos processos em execução.

```
/* Obtenção de um objecto WbemLocator e estabelecimento da   */
/* igação à máquina remota $host para o namespace $namespace */
$wbem = Win32::OLE->new("WbemScripting.SWbemLocator");
$wbemServices = $wbem->ConnectServer($host,$namespace);

/* Execução da query: todas as instâncias da classe Win32_Process */
$query = "SELECT * FROM Win32_Process";
$objects = $wbemServices->execquery($query);

/* Processamento do resultado da query. É feito o   */
/* print de alguns campos das instâncias retornadas */
foreach $object (in($objects)) {
    print $object->name;              // nome do processo
    print $object->description;       // descrição do processo
    print $object->handle;            // handle do processo
}
```

Figura 32 – Código Perl para acesso remoto WMI

O código da figura 32 está dividido nos três blocos referidos no parágrafo anterior. O estabelecimento da ligação à máquina remota é feito através da criação de um objecto

WbemScripting.SWbemLocator, seguido da conexão ao *host* e ao *namespace* desejado. Depois de estabelecida a ligação é executada a *query* e processados os resultados da mesma. No exemplo apresentado pretende-se obter uma listagem dos processos, fazendo o *print* do nome, da descrição e do *handle* de cada um dos processos retornados.

Além do acesso WMI e do uso de Perl em *scritps* para consulta de dados, uma das principais utilizações do Perl no desenvolvimento da plataforma de gestão foi na criação e gestão de tarefas automáticas. Para tal, foi utilizado um módulo (`Win32::TaskScheduler`) que integra com o sistema operativo, permitindo qualquer tipo de operação na gestão das tarefas. Este módulo apresenta-se como uma interface em Perl entre o utilizador e a API do sistema operativo, permitindo ao utilizador interagir directamente com as tarefas automáticas do sistema operativo.

Por exemplo, a criação de uma tarefa automática que execute apenas uma vez, segundo uma determinada data e hora definida pelo técnico torna-se tão simples como o extracto de código da figura 33 apresenta.

```
/* Indicação ao Perl para usar o módulo WIN32::TaskScheduler  */
use Win32::TaskScheduler;

/* Inicialização de um objecto sobre o qual será feito o acesso   */
/* à API do sistema operativo para a criação da tarefa automática */
$scheduler = Win32::TaskScheduler->New();

/* Configuração temporal da tarefa: é definido uma hora e uma  */
/* data a que a tarefa se iniciará. É configurado ainda o tipo */
/* de tarefa através de uma constante pré-definida (na API):   */
/* TASK_TIME_TRIGGER_ONCE (executar apenas uma vez)            */
%trigger = &createTrigger($startTime,$startDate);
$trigger{'TriggerType'} = $scheduler->TASK_TIME_TRIGGER_ONCE;

/* Alocação do trigger (configuração temporal) a uma nova tarefa */
$scheduler->NewWorkItem($task,\%trigger);

/* É definida a aplicação que será lançada pela */
/* tarefa bem como o directório de trabalho.    */
$scheduler->SetApplicationName($app);
$scheduler->SetWorkingDirectory($workDir);

/* Após a operação de Save(), a tarefa é realmente guardada   */
/* pelo sistema operativo e o objecto $scheduler é libertado. */
$scheduler->Save();
```

Figura 33 - Código Perl para criação de tarefas automáticas (execução apenas uma vez)

No Anexo C são apresentados exemplos para outras periodicidades na configuração das tarefas.

4.4.3 Framework de Apresentação - PRADO

Tratando-se de um sistema com uma filosofia cliente/servidor, para o desenvolvimento do subsistema de apresentação da plataforma de gestão existiam inicialmente duas opções: desenvolver uma aplicação de *desktop* ou uma aplicação Web. Dada a tendência actual para a *webização* de vários processos e aplicações, a escolha recaiu sobre a segunda opção. As vantagens são nítidas, desde a centralização do desenvolvimento, sem a necessidade de *software* extra no lado do cliente (basta um simples *browser*) até à portabilidade do código de apresentação entre várias plataformas.

Neste sentido, procurou-se numa fase inicial, seleccionar um *framework* de base, preferencialmente *open-source*, a partir do qual se pudesse proceder ao desenvolvimento da plataforma. Os requisitos que o *framework* deveria respeitar eram a capacidade de instanciação de objectos COM (essencial no acesso WMI) bem como o suporte de bases de dados relacionais. Outros aspectos de particular interesse seriam a programação orientada a objectos e a simplificação no tratamento de eventos (e.g. processamento de *forms*).

Na altura em que foi necessário escolher uma plataforma de desenvolvimento, existiam alguns *frameworks* disponíveis: QCODO [60], PHOCOA [61], PRADO [56], CAKEPHP [62] e WASP [63]. Cada um deles foi analisado, no sentido de satisfazerem ou não os requisitos que eram necessários para o desenvolvimento da plataforma de gestão, enumerados no parágrafo anterior. Alguns deles apresentavam a desvantagem de dependerem de *packages* adicionais como o caso do PHOCOA e do WASP que requeriam algum *software* PHP extra para a compilação de projectos e gestão de *layouts* das páginas. O CAKEPHP era, na altura, um *framework* algo incompleto, ainda numa fase embrionária de desenvolvimento e cuja dependência sobre o modelo de dados da aplicação a desenvolver era algo a evitar. Neste mesmo ponto, a plataforma QCODO era também demasiado dependente do modelo de dados. Aliás, o funcionamento da plataforma QCODO consiste precisamente na geração de código automaticamente, a partir do modelo de dados. Além disso, o número de componentes de base em qualquer das plataformas (excepto para o PRADO) era extremamente reduzido e cingiam-se quase exclusivamente aos componentes de formulários. Exceptuando a plataforma QCODO, com suporte apenas para o sistema de base de dados MySQL, todas as outras plataformas dispunham de classes de abstracção no acesso às bases de dados, umas por PROPEL [64] (PHOCOA), outras por ADODB [65] (PRADO, CAKE) e outra pelo PEAR DB [66] (WASP). Todas elas estavam assentes na versão 5 do PHP, à excepção do CAKE que, além da versão 5, era compatível com a versão 4 do PHP. Um outro ponto também analisado foi a necessidade de instalação do *framework* no sistema operativo. Relativamente a este item,

apenas o PRADO e o CAKE não requerem qualquer tipo de instalação bastando copiar o conteúdo do *framework* para o local desejado no *filesystem*.

Após a análise às plataformas descritas, a escolha recaiu sobre o PRADO por vários motivos: possuía um leque bastante alargado de componentes predefinidos, simplicidade no suporte de módulos permitindo uma melhor organização do código fonte; não necessita de qualquer tipo de instalação podendo funcionar em qualquer sistema operativo onde o PHP seja interpretado; independência em relação ao sistema de base de dados; programação orientada por objectos com PHP 5.

PRADO é o acrónimo de ***P**HP **R**apid **A**pplication **D**evelopment **O**bject-oriented*. Trata-se de um *framework* para PHP5 baseado em componentes e orientado a eventos para o desenvolvimento de aplicações *web*. A primeira versão do PRADO surgiu em Junho de 2004, escrita em PHP4. No entanto, a sua popularidade cresceu exponencialmente quando venceu o concurso promovido pela Zend [68] para aplicações desenvolvidas em/para PHP5, tendo nessa altura migrado para a versão 5 de PHP.

As grandes vantagens do uso do PRADO são:

- Reutilização de código – o uso de componentes proporciona um elevado nível de reutilização de código
- Simplicidade – apesar de discutível no primeiro impacto, o PRADO revela-se uma plataforma extremamente intuitiva e flexível. Possui internamente um leque infindável de funcionalidades que cobrem praticamente todas as necessidades ao nível das aplicações Web.
- Robustez – a sua filosofia de programação orientada a objectos liberta o programador da escrita das partes de código mais aborrecidas. Como é baseado em PHP5, usa o novo mecanismo de excepções que permite um *reporting* de erros bastante mais preciso.
- Performance – graças a técnicas de *caching*, as aplicações PRADO são extremamente rápidas.
- Modularidade – PRADO separa claramente os componentes de conteúdo e apresentação das páginas e a lógica associada às mesmas. O conceito de módulos permite também a divisão de tarefas entre vários elementos na fase de desenvolvimento.

O desenvolvimento de uma aplicação *web* em PRADO consiste num conjunto de páginas com funcionalidades específicas dessa própria aplicação. De certa forma, o mesmo se passa em qualquer outro tipo de *framework* para desenvolvimento *web*. Contudo, é na forma como essas páginas são construídas e apresentadas ao utilizador que reside a diferença entre esta e outras plataformas. Em PRADO, qualquer página é constituída por um *template* (página que contém essencialmente código HTML para apresentação no *browser*) e uma classe

(ficheiro de código onde reside toda a lógica associada à página). Para facilitar a construção das páginas e a sua apresentação, o PRADO disponibiliza um conjunto de componentes que permitem ao programador simplificar o código HTML e o tratamento de eventos gerados pelos componentes. Por exemplo, um botão para submissão de um formulário é um componente PRADO ao qual se pode associar um evento, despoletado pelo acto de submissão por parte do utilizador. Na realidade, uma aplicação em PRADO consiste num conjunto de componentes dado que uma página em PRADO é em si própria um componente.

Nas secções que se seguem será feita uma breve introdução aos componentes PRADO e ao modo como se pode construir uma simples aplicação em PRADO. Vejamos então o que é um componente PRADO e pelo que é constituído.

4.4.3.1 Componentes PRADO

Um componente PRADO é uma combinação de um ficheiro de especificação XML e uma classe PHP. Enquanto que a especificação XML consiste na declaração das propriedades e eventos associados ao componente, na classe são implementados os respectivos mecanismos e lógica associada ao componente. Na prática, o ficheiro de especificação é como se uma apresentação da classe onde se declaram as variáveis públicas da mesma (as propriedades do componente). Além das propriedades, a especificação do componente pode conter também a declaração de eventos. Os eventos são os nomes dados aos métodos que serão posteriormente responsáveis pelo tratamento desses eventos. Consideremos o exemplo seguinte para o componente `TFileUpload`. Trata-se de um componente usado para o *upload* de ficheiros. A figura 34 apresenta a especificação do componente.

TFileUpload.spec

```
<?xml version="1.0" encoding="UTF-8"?>
<component>
	<property name="LocalName" get="getLocalName" type="string" />
	<property name="FileName" get="getFileName" type="string" />
	<property name="FileSize" get="getFileSize" type="string" />
	<property name="FileType" get="getFileType" type="string" />
	<property name="UploadError" get="getUploadError" type="string" />
	<property name="Uploaded" get="getUploaded" type="string" />
	<property name="MaxFileSize" get="getMaxFileSize" set="setMaxFileSize"
		type="string" />

	<event name="OnFileUpload" />
	<event name="OnFileUploadFailed" />
</component>
```

Figura 34 – Especificação XML de um componente PRADO

O componente `TFileUpload` tem sete propriedades: `LocalName`, `FileName`, `FileSize`, `FileType`, `UploadError`, `Uploaded`, `MaxFileSize`. Dependendo da funcionalidade do componente, as suas propriedades podem ser lidas ou escritas ou seja, para cada propriedade poderão haver métodos de escrita (`set`) e/ou de leitura (`get`). Trata-se de métodos da classe que são públicos e que permitem aceder às variáveis da classe (propriedades). No exemplo apresentado, à excepção da propriedade `MaxFileSize`, para todas as outras o componente disponibiliza apenas métodos de leitura: `getLocalName`, `getFileName`, `getFileSize`, `getFileType`, `getUploadError`, `getUploaded`. Para a propriedade `MaxFileSize`, além do método de leitura `getMaxFileSize` é também possível definir o valor dessa propriedade através do método `setMaxFileSize`. As propriedades contêm ainda a definição de tipo, sendo que no exemplo apresentado todas são do tipo *string*. Além das propriedades, a utilização do componente pode gerar dois tipos de eventos, os quais se encontram declarados na especificação (obrigatório): `OnFileUpload`, `OnFileUploadFailed`. Estes eventos são gerados (ou podem ser gerados) por acção do utilizador/visualizador (o *upload* do ficheiro). A ocorrência do evento `OnFileUpload` indica que o *upload* do ficheiro foi efectuado com êxito enquanto que a ocorrência do evento `OnFileUploadFailed` é indicadora de erro.

Ainda em relação à especificação dos componentes e apesar de não ser visível neste exemplo, as propriedades de um componente podem ter definidos valores por omissão, sendo tal indicado, por exemplo, por `default="My Name"` para uma propriedade do tipo *string*.

A lógica associada ao componente `TFileUpload` reside na classe (ficheiro `TFileUpload.php`) que, por questões de simplicidade e por não acrescentar informação útil ao exposto, não é apresentada.

4.4.3.2 Aplicação PRADO

O desenvolvimento de uma aplicação em PRADO envolve a combinação de vários componentes e código HTML para a criação e apresentação das páginas da aplicação. Além das páginas que constituirão a aplicação, são ainda necessários dois ficheiros específicos: um deles como ponto de entrada da aplicação e outro ficheiro de especificação (em XML) como veremos no exemplo apresentado adiante.

A título de exemplo, e aproveitando parte do código desenvolvido para o sistema de autenticação da plataforma de gestão, apresenta-se a seguir uma pequena aplicação constituída por apenas duas páginas: `LoginPage` e `LoggedInPage`. A primeira (`LoginPage`) é a *default page* da aplicação ou seja, aquela para onde o utilizador é imediatamente redireccionado quando acede ao *site*. A outra (`LoggedInPage`), de acesso restrito (por autenticação na primeira), disponibiliza ao utilizador apenas a funcionalidade de

se desautenticar, redireccionando o utilizador para a página `LoginPage`. Os ficheiros que constituem esta simples aplicação são, portanto, os seguintes:

– `index.php`, página de entrada da aplicação;

– `exampleApp/application.spec`, ficheiro de especificação da aplicação;

– `exampleApp/LoginPage.php`, classe da página com formulário de autenticação (*default page*);

– `exampleApp/LoginPage.tpl`, *template* da página com formulário de autenticação (*default page*);

– `exampleApp/LoggedInPage.php`, classe da página de acesso privilegiado;

– `exampleApp/LoggedInPage.tpl`, *template* da página de acesso privilegiado;

A página de entrada da aplicação, apresentada na figura 35 consiste apenas na indicação da localização do *framework* (função `require_once` com indicação da *path* relativa onde se encontra) e no lançamento da aplicação, invocando uma função específica do *framework* (`pradoGetApplication('exampleApp/LoginPage.php')->run()`), á qual é passado como parâmetro a *default page* da aplicação.

index.php

```
<?php
require_once('../framework/prado.php');
pradoGetApplication('exampleApp/LoginPage.php')->run();
?>
```

Figura 35 – Página de lançamento de uma aplicação PRADO

O ficheiro de especificação da aplicação consiste num conjunto de parâmetros específicos da aplicação, escritos num documento XML, conforme a figura 36.

exampleApp/application.spec

```
<?xml version="1.0" encoding="UTF-8"?>
<application ID="ExampleApp">

  <request default="LoginPage" />
  <user class="UsersClass" />

  <alias name="Pages" path="." />
  <using namespace="System.Web.UI.WebControls" />
  <using namespace="Pages" />
  <secured page="LoggedInPage" />

  <parameter name="DSN">mysql://user:password@host/database</parameter>

</application>
```

Figura 36 – Documento XML de especificação de uma aplicação PRADO

Nas especificações da aplicação, encontramos vários elementos cujas propriedades definem o modo como a aplicação é executada. O elemento `request`, através do seu atributo `default` define a página que deverá ser carregada por defeito pelo cliente caso não seja especificada nenhuma. O elemento `user` define a classe que será usada para o tratamento dos utilizadores e dos processos de autenticação e autorização. Esta directiva é usada apenas em aplicações onde é necessário recorrer a mecanismos de autenticação e autorização. Para o exemplo apresentado, é especificado que a aplicação irá recorrer à classe `UsersClass`, classe essa predefinida no *framework,* que contém já alguma lógica associada ao processamento da autenticação e autorização de utilizadores. Contudo, se for necessário, o programador pode definir uma classe própria para a gestão de utilizadores ou simplesmente estender a classe nativa do *framework*. A directiva `alias` define *path aliases*. O sistema de `alias` interno do *framework* é definido de modo que qualquer referência seja feita em relação ao directório contendo o código do *framework*. O uso de `alias` permite a criação de estruturas de directórios físicos com vários níveis ou num outro ponto do *filesystem* (normalmente inacessível pelo *framework*) sendo possível ao programador chamá-los na aplicação através do seus atalhos, os `alias`. A directiva `using` permite especificar quais os `alias` que serão usados quando a aplicação for iniciada. Para que a segunda página (`LoggedInPage`) seja uma página protegida é necessário declará-la como tal no ficheiro de especificação da aplicação. Deste modo, a página só será carregada do lado do cliente caso este esteja autenticado. Caso contrário será redireccionado para a *default page,* indicada no atributo `default` da directiva `request`. Por último, o ficheiro de especificações da aplicação contém ainda um parâmetro com o nome DSN e cujo conteúdo é a *string* de ligação a uma base de dados MySQL. Dado que as ligações a bases de dados são normalmente únicas para qualquer conjunto de páginas de uma aplicação, é boa prática colocar o parâmetro associado à aplicação (e não a uma única página) de forma que qualquer página (e respectiva classe) possa usar este parâmetro caso necessite de aceder à referida base de dados.

Os restantes ficheiros (`LoginPage.php`, `LoginPage.tpl`, `LoggedInPage.php`, `LoggedInPage.tpl`) contêm a lógica e a apresentação das páginas que constituem esta aplicação.

Vejamos em primeiro lugar a página `LoginPage`. Esta é constituída pelo *template* HTML (ficheiro com extensão `.tpl`) e pela classe (ficheiro com extensão `.php`). Como já foi dito anteriormente, o *template* contém o *layout* da página ou seja, a forma como a informação será apresentada. Para a apresentação da informação, o programador pode recorrer a simples HTML ou embeber componentes PRADO no HTML da página (como se verá no exemplo apresentado). Na classe reside a lógica associada às funcionalidades da página e aos

componentes da mesma (caso a página contenha componentes). Vejamos então, primeiro, o *template* (figura 37).

```
exampleApp/LoginPage.tpl

<HTML>
<HEAD>
</HEAD>

<BODY>

<com:TForm>

  <p>Para ter acesso a esta área do site, p.f., introduza o username e
password</p>
  <table border="0">
    <tr>
      <td>Username</td>
      <td><com:TTextBox ID="User" /></td>
    </tr>
    <tr>
      <td>Password</td>
      <td><com:TTextBox ID="Pass" TextMode="Password" /></td>
    </tr>
    <tr>
      <td></td>
      <td><com:TButton ID="button" Text="Login" OnClick="onLogin" /></td>
    </tr>
  </table>
</com:TForm>

</BODY>
</HTML>
```

Figura 37 – *Template* de uma página PRADO com componentes embebidos

No código apresentado na figura 37, identifica-se claramente os componentes PRADO (texto de cor vermelha). Tratam-se dos componentes: `TForm`, `TTextBox` e `TButton`. O primeiro é o indicador de formulário. Dado que se trata de uma página para autenticação com caixas de texto e um botão de submissão, é necessário que haja um formulário (tal como o uso das *tags* `<form>` e `</form>` em HTML). `TTextBox` é o componente para caixas de texto e `TButton` o componente para o botão de submissão. Apesar de opcional, os componentes tem normalmente um identificador `ID` (que deverá ser único na página) e um conjunto de propriedades. No caso do componente `TTextBox`, a propriedade `TextMode` definida como `Password` indica que a caixa de texto é do tipo *password* enquanto que no componente `TButton` a propriedade `Text` define o texto que aparecerá no botão. Note-se

ainda a propriedade `OnClick` que associa um *event handler* `onLogin` à operação de submissão do formulário pelo botão.

A classe para esta página apresenta-se na figura 38. Note-se que a associação entre a classe e o *template* é feita pelo nome: `LoginPage`. Além do nome da página, este deve também ser o nome da classe.

exampleApp/LoginPage.php

```
<?php

class LoginPage extends TPage
{

  public function onLogin($sender,$param) {
     if($this->Username->Text == 'admin' && $this->Pass->Text == 'admin')) {
        $this->User->Login->($this->Username->Text);
        $this->Application->transfer('LoggedInPage');
     }
     else
       $param->isValid=false;
  }
}
?>
```

Figura 38 – Classe de uma página PRADO com um *event handler*

O código da figura 38 consiste na definição de uma classe de nome `LoginPage` que contém apenas um método: `onLogin`. Este método é o *event handler* associado à submissão do botão com o ID `button` no *template* (figura 37). Este método apenas valida o utilizador de acordo com o *username* e *password* (testando-os contra o par `admin/admin`) redireccionando-o para a página de acesso restrito (`LoggedInPage`) apenas no caso de a autenticação ser bem sucedida. Caso contrário, o utilizador permanece na mesma página. Note-se ainda o seguinte código: `$this->User->Login($this->Username->Text)`. Foi já dito anteriormente (na descrição da especificação da aplicação) que o PRADO disponibiliza uma classe (`UsersClass`) para a gestão de utilizadores. Feita a verificação das credenciais do utilizador através do *event handler* `onLogin`, é necessário indicar à classe que o utilizador corrente se encontra autenticado.

A conjunção dos dois ficheiros apresentados, o *template* e a classe, disponibilizará ao utilizador algo semelhante ao apresentado na figura 39 na página seguinte.

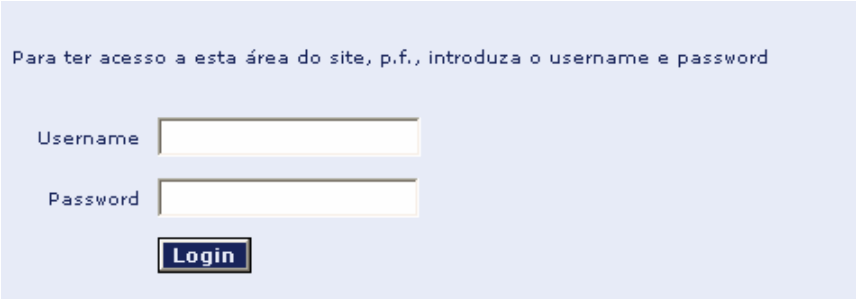


Figura 39 – Página de *Login* em PRADO

Depois de autenticado, o utilizador terá então acesso à segunda página que, tal como a primeira, consiste num *template* e numa classe. O resultado será algo semelhante à figura 40.

Figura 40 – Página de *logout* em PRADO

Além da informação mostrada, é disponibilizado ao utilizador um botão para que este possa terminar a sua sessão. O *template* correspondente ao *layout* apresentado na figura 40 encontra-se na figura 41.

exampleApp/LoggedInPage.tpl

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<com:TForm>
  <p>Acabou de se autenticar</p>
  <table border="0">
    <tr>
      <td>Username</td>
      <td> <%=$this->User->getUsername()%> </td>
    </tr>
    <tr>
      <td></td>
      <td><com:TButton ID="button" Text="Logout" OnClick="onLogout" /></td>
    </tr>
  </table>
</com:TForm>
</BODY>
</HTML>
```

Figura 41 – *Template* de uma página PRADO

Em relação ao *template* importa introduzir uma pequena nota. Parte da informação disponibilizada nesta página corresponde ao *username* do utilizador corrente. Isto é feito introduzindo código php no *template* mas de uma forma característica do PRADO: através das *tags* <% e %> com o conteúdo `%this->User->getUsername()`. O *username* é então acedido da seguinte forma: `$this` representa a página em causa; o objecto `User` é persistente na aplicação; `User` disponibiliza o método `getUserName()` que permite ler a sua variável `username`.

A classe correspondente ao *template* apresentado consiste no seguinte código, apresentado na figura 42.

exampleApp/LoggedInPage.php

```
<?php

class LoggedInPage extends TPage
{

  public function onLogout($sender,$param) {
       $this->User->logout();
       $this->Application->transfer();
  }
}
?>
```

Figura 42 – Classe de uma página em PRADO

O *event handler* responsável pela operação de *logout* apenas indica à classe `User` que o utilizador fez *logout* (invocação do método `logout()` da classe `User`) e redirecciona-o para a *default page* da aplicação (por omissão na invocação do método `transfer()` da aplicação), definida no ficheiro de especificação da mesma.

5 Resultados

Neste capítulo são apresentados os resultados obtidos com a plataforma de gestão desenvolvida. A plataforma foi testada com um conjunto de máquinas representando um parque informático. Nos subcapítulos que se seguem serão apresentadas com algum detalhe as funcionalidades disponibilizadas pela plataforma de gestão, acompanhadas de alguns *screenshots*. A abordagem será do tipo *quick-start tutorial*, mostrando as várias funcionalidades da plataforma sem se tornar demasiado exaustiva. Além de uma breve introdução, a descrição será feita por módulos, na seguinte ordem:

- gestão de utilizadores
- gestão de classes CIM e extensões
- gestão de máquinas
- monitorização online
- gestão de tarefas automáticas
- monitorização offline
- alertas e notificações

5.1 Introdução

A plataforma de gestão centralizada de parques informáticos foi desenvolvida em um ambiente *web* por vários motivos e que foram já expostos no capítulo anterior. Como tal, o sistema apresenta-se ao utilizador como um conjunto de páginas organizadas por tipo de funcionalidade ou seja, por módulos. Para que o administrador do sistema ou os técnicos possam operar sobre ele devem proceder à respectiva autenticação, caso contrário, apenas tem acesso a uma breve apresentação do projecto e alguma documentação. Em qualquer dos casos, a navegação no site é feita recorrendo a um sistema de menus que se abrem de acordo com o tipo de utilizador e o respectivo nível de autorização. Estes menus situam-se no topo da página sendo portanto de fácil acesso. A figura 43 apresenta a página de entrada do *site* e os menus disponíveis para qualquer utilizador antes do processo de autenticação.

Figura 43 – Página de apresentação da plataforma de gestão

Após a autenticação, os técnicos tem acesso a um menu que varia de acordo com o tipo de autorização. Por exemplo, o(s) administrador(es) do sistema tem acesso a um módulo invisível para os restantes técnicos: o de gestão de utilizadores. Um exemplo do menu apresentado ao administrador, com detalhe sobre as funcionalidades de gestão de utilizadores apresenta-se na figura 44.

Figura 44 – Menu de opções da plataforma de gestão

Quanto às restantes opções do menu, correspondem às diversas funcionalidades disponibilizadas:

– Geral: informação geral sobre o projecto e o trabalho desenvolvido

– Máquinas: acesso ao módulo de gestão de máquinas

– Classes: acesso ao módulo de gestão de classes, propriedades e métodos

– Gestão onLine: funcionalidades para gestão das máquinas em tempo real

– Tarefas: configuração e gestão de tarefas automáticas para recolha de informação das máquinas geridas

– Relatórios: configuração e visualização de relatórios sobre configurações e estados das máquinas e estado do parque informático

– Alertas: acesso ao módulo de gestão de alertas e notificações

Estas funcionalidades serão analisadas nas secções que seguem.

5.2 Gestão de Utilizadores

O módulo de gestão de utilizadores da plataforma, designados doravante por técnicos, tal como o próprio nome indica, tem como principal objectivo a segurança da gestão ou seja, autorizar o acto e/ou as tarefas de gestão apenas aos utilizadores com permissões e autorização para tal.

Sendo um dos objectivos deste trabalho demonstrar a utilidade da centralização da gestão de um parque informático, não menos importante é a capacidade da plataforma central de gestão ser utilizada de forma descentralizada.

Por exemplo, para um parque informático distribuído por vários escritórios ou localizações geográficas dispersas, é de todo conveniente possibilitar que o acto de gestão seja praticado, dependendo obviamente do nível de autorização, por uma equipa de técnicos. Na realidade, para parques informáticos de grande dimensão, seria impensável "entregar" o acto de gestão a um único administrador. Na prática, a gestão acaba por ser distribuída por uma equipa de administração, preferencialmente dispersa geograficamente, de modo a cobrir as várias localizações físicas da instituição/organização. Consequentemente, o nível de gestão atribuído a cada um dos membros da equipa de gestão depende do nível de autorização associado a esse técnico.

Pegando num caso concreto, ao nível das funcionalidades disponibilizadas pela plataforma de gestão, pode dar-se o caso de um técnico ter permissões para monitorizar em tempo real um conjunto de máquinas mas sem poder criar tarefas ou configurar alertas relativamente a esse conjunto.

Após autenticação, cada técnico tem acesso a uma página de perfil, cujo acesso pode ser feito através da opção no menu (figura 44) correspondente, que lhe permite além da visualização de informação pessoal, efectuar operações como a mudança de *password* (figura 45).

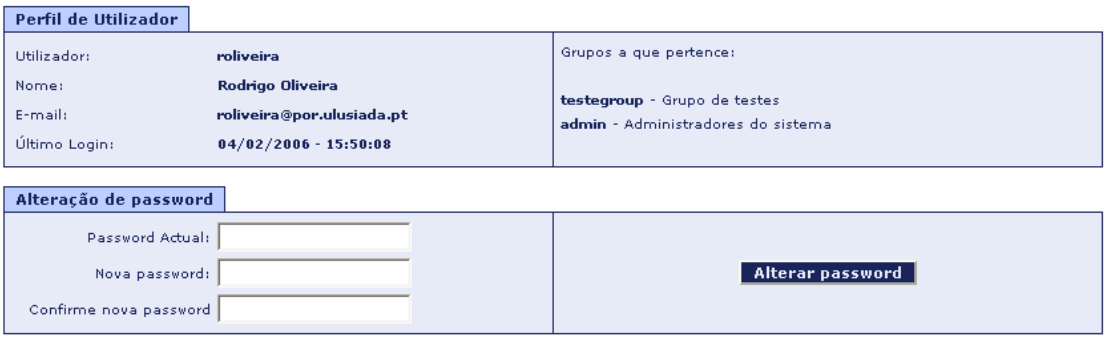


Figura 45 – Perfil de utilizador (técnico) da plataforma de gestão

A funcionalidade de gestão de utilizadores, acedida via opção `Gestão de Utilizadores` (ver figura 44), permite a criação de utilizadores (técnicos), associação a

grupos e alteração dos níveis de autorização. Esta gestão é feita por um utilizador privilegiado (administrador) o qual possui, obviamente, um nível de autorização máximo. A figura 46 representa uma listagem de utilizadores da plataforma e respectiva edição.

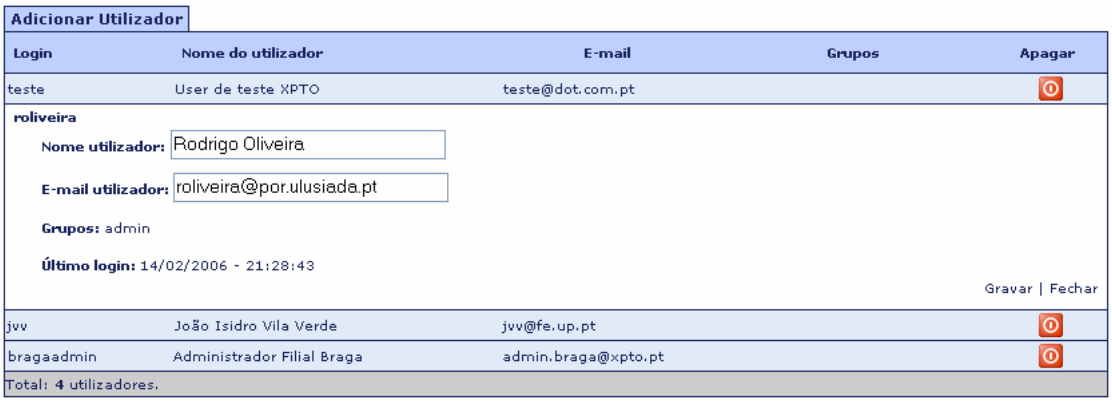


Figura 46 – Administração de utilizadores do sistema

Relativamente à gestão dos grupos, esta é feita através de uma outra interface apresentada na figura 47. É também através desta interface que são associados utilizadores do sistema a grupos. Da lista de utilizadores disponíveis, o administrador do sistema pode associá-los aos grupos criados. Neste aspecto, é permitido que um utilizador possa pertencer a mais do que um grupo.

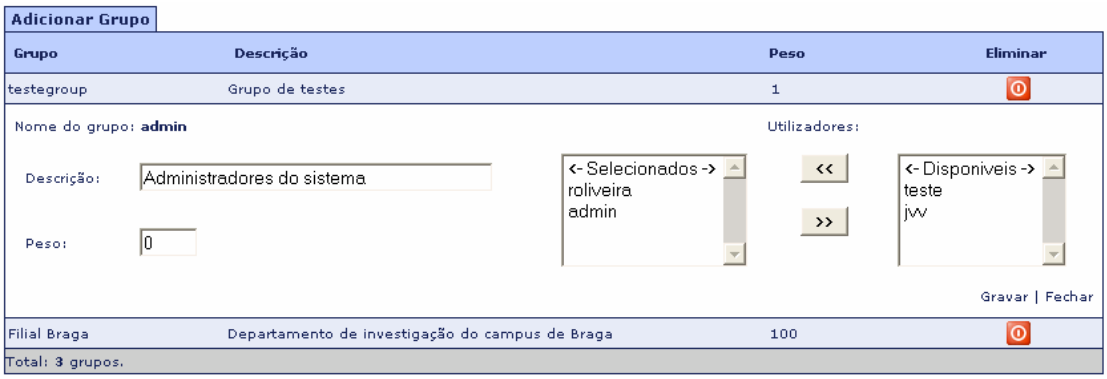


Figura 47 – Gestão de grupos de utilizadores do sistema

Ainda relativamente aos grupos de utilizadores, estes são também caracterizados por um atributo "peso" que permitirá atribuir níveis de autorização aos utilizadores de acordo com os grupos a que pertencem (ver secção **4.3.1**).

5.3 Gestão de Classes CIM e Extensões

O módulo de gestão de classes foi desenvolvido tanto para documentação e consulta como para a configuração avançada de tarefas automáticas (secção **5.6**). O acesso às opções

para gestão de classes, propriedades e métodos é feito através da barra de menu, conforme a figura 48.

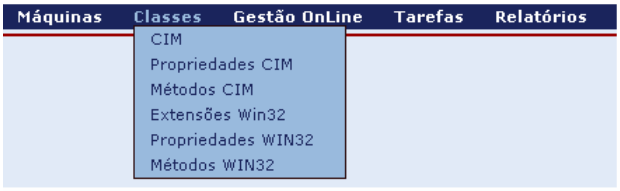


Figura 48 – Acesso ao módulo de gestão de classes

Apesar de assente numa estrutura de dados única, em termos práticos são apresentadas ao administrador duas interfaces idênticas: uma para a gestão de classes CIM e outra para a gestão de extensões do modelo. Estas interfaces apresentam ao administrador a descrição das classes de maior relevo no âmbito da gestão de parques informáticos bem como das respectivas propriedades e métodos. Na figura 49 está representada a listagem de propriedades para a classe `CIM_OperatingSystem` do esquema CIM.

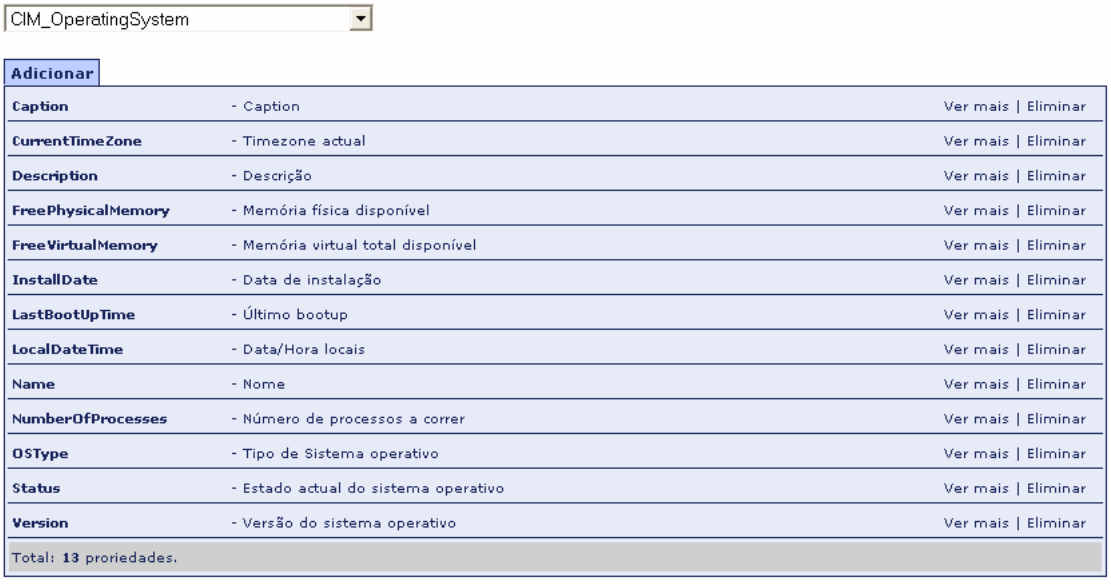


Figura 49 – Listagem de propriedades para uma classe do esquema CIM

A gestão das classes, propriedades e métodos é da exclusividade do administrador da plataforma sendo que os técnicos apenas podem visualizar a informação e não editá-la.

Quanto às extensões ao modelo CIM, a outra interface, em tudo idêntica à anterior, contém as classes relativas às extensões `WIN32` ou seja, as extensões desenvolvidas pela Microsoft para a gestão dos seus sistemas operativos.

Por último, a listagem de métodos, apresentada na página seguinte na figura 50 para a classe `Win32_NetworkAdapterConfiguration`, é em tudo idêntica à listagem de propriedades das classes, sendo possível para cada um dos métodos da classe visualizar informação mais detalhada bem como os parâmetros do mesmo.

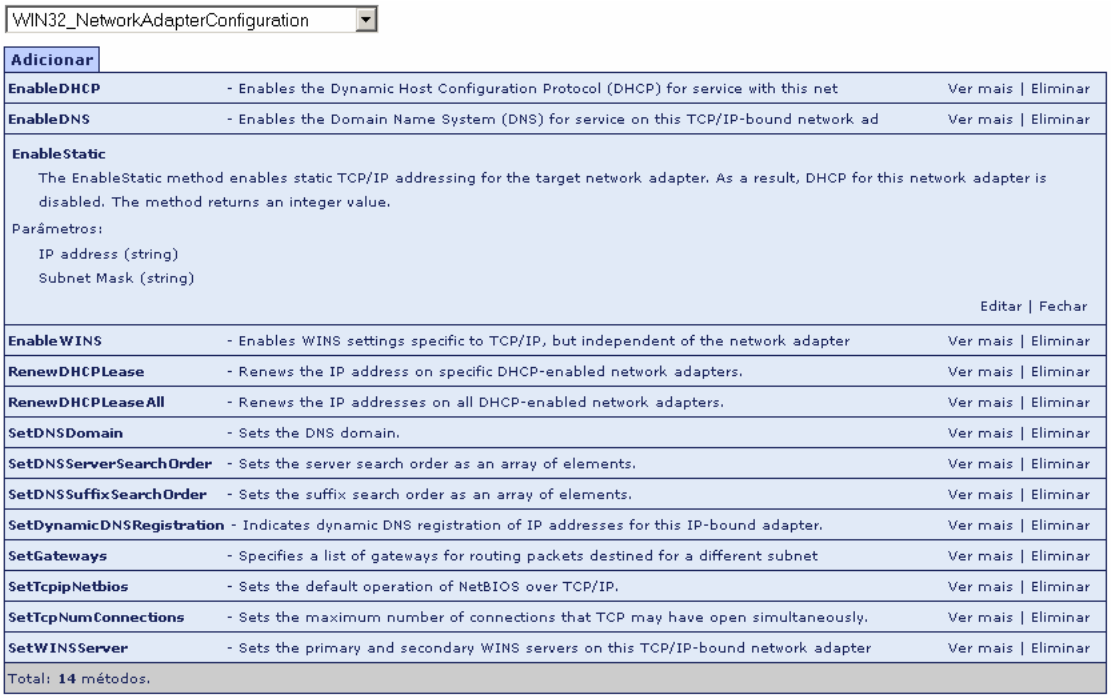


Figura 50 – Listagem de métodos para uma classe Win32

No exemplo apresentado, o método `EnableDNS` tem dois parâmetros: o endereço IP e a máscara de rede.

5.4 Gestão de Máquinas

O módulo de gestão de máquinas tem como principal objectivo gerir o universo de máquinas que fazem parte do parque informático e a informação básica ao nível individual. Aqui, o técnico pode adicionar ou remover máquinas ou simplesmente actualizar algumas características das mesmas. Além disso, o técnico pode ainda obter alguns relatórios sobre o histórico das máquinas e do *hardware* que as compõe. O acesso a estas funcionalidades é feito através da respectiva opção no menu conforme a figura 51.

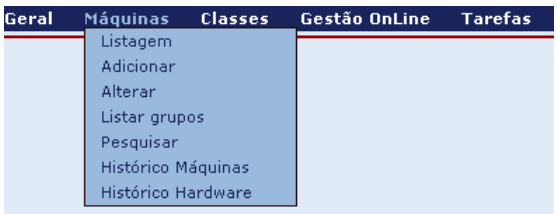


Figura 51 – Acesso às funções de gestão de máquinas

As funcionalidades disponibilizadas incluem a listagem das máquinas pertencentes ao universo de gestão, a adição e edição de máquinas, a gestão de grupos, um motor de pesquisa de máquinas e/ou de componentes de *hardware* de acordo com vários critérios e ainda a elaboração de alguns relatórios acerca do histórico das máquinas e do *hardware* que compõem o universo de gestão. A figura 52 na página seguinte apresenta a interface de actualização/edição de máquinas.

Figura 52 – Informação genérica sobre uma máquina

Relativamente a esta figura e respectiva funcionalidade, importa referir os dois modos como o técnico a pode usar. Por um lado, serve como interface de adição de novas máquinas ao sistema de gestão. Neste caso, o técnico poderá preencher as caixas de texto com o endereço IP, o nome DNS e o nome do computador (vulgo *netbios*) e definir o estado da máquina em termos de gestão (`ON`, `OFF`, `TESTING`). Por outro lado, caso não disponha da informação necessária ou pretenda obter a informação corrente na máquina alvo, o técnico deverá apenas introduzir o endereço IP e ler os "`Valores actuais na máquina`". A informação relativa ao sistema operativo, a data de inserção no sistema e o técnico que o fez aparece apenas nos casos em que se edita uma máquina que já faz parte do universo de gestão. O mesmo acontece relativamente aos grupos a que a máquina pertence, opção disponibilizada através do menu em `Máquinas->Gestão de grupos`. A informação de localização, opcional, deverá ser preenchida pelo técnico. Além da identificação da máquina através do seu endereço, nome, sistema operativo, etc., foram criados outros atributos que permitem classificar as máquinas em termos das suas localizações. Para este efeito foram considerados os seguintes atributos: unidade, departamento, edifício e cidade. Esta informação permitirá aos técnicos a criação de grupos de máquinas de acordo com estes critérios funcionais. Por exemplo, poderá existir um grupo de máquinas contendo todas as máquinas de um edifício numa cidade ou então, as máquinas de um departamento, independentemente do edifício ou cidade. A "`Actualização da Base de Dados`" deverá ser feita sempre que sejam alteradas algumas destas características da máquina.

Outra característica importante na identificação da máquina é o seu *hardware*. O levantamento do principal *hardware* que a constitui (componentes) como discos, processador, memórias, placa gráfica, etc., é usado directamente na identificação da máquina. A informação relativa ao *hardware* das máquinas aparece na mesma página que a informação apresentada na figura 52 estando disponível uma *combo box* para a escolha do tipo de componente a consultar. Na página seguinte, a figura 53 apresenta uma listagem das características associadas ao componente `Adaptador de Rede` de uma máquina.

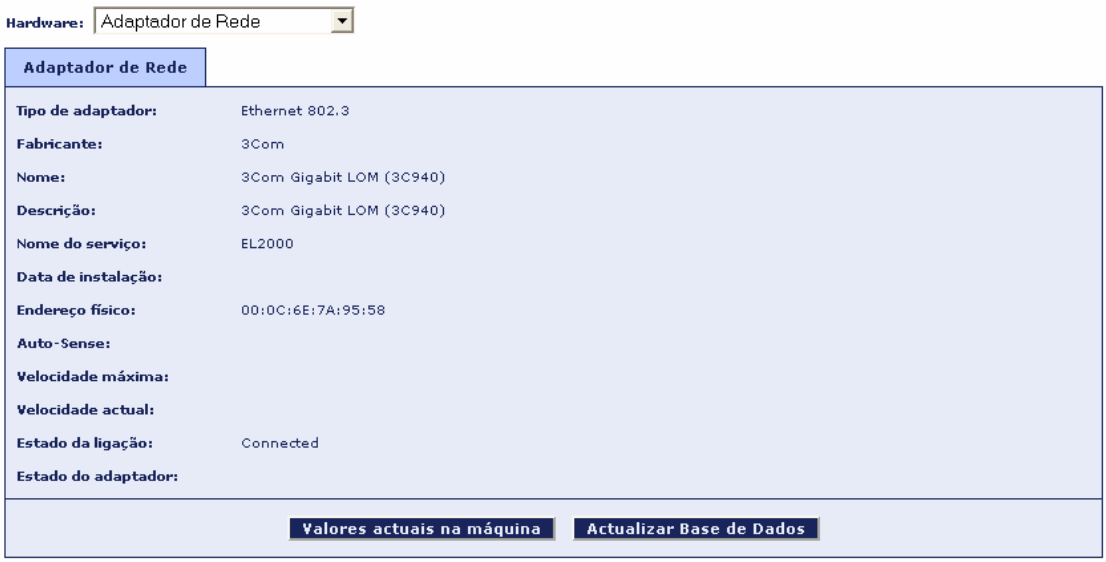


Figura 53 – Informação específica sobre o adaptador de rede de uma máquina

Para a consulta do *hardware* das máquinas são disponibilizadas as seguintes categorias: processador, *motherboard*, BIOS, leitor CD/DVD, memória, placa gráfica, adaptador de rede, placa de som, disco rígido, teclado, rato. A informação apresentada na figura 53 para o componente `Adaptador de Rede` consiste num conjunto de características específicas do componente, algumas delas com valores definidos, outras em branco. Sendo a informação proveniente do sistema de informação, é possível recolhê-la directamente a partir da máquina em tempo real, podendo o técnico posteriormente actualizar a base de dados.

De facto, um dos problemas que se coloca na administração de um parque informático é precisamente o de saber quem trocou o quê e quando. Suponhamos que, num local remoto, uma avaria de um disco numa máquina leva à sua substituição por outro de igual capacidade mas de um fabricante diferente. Dado que nem sempre estas situações são comunicadas à equipa de gestão informática, torna-se relativamente simples detectar o caso através de leituras periódicas da informação dos componentes de uma máquina. Dado que foram definidas estruturas de dados capazes de armazenar a informação ao nível individual dos componentes que constituem uma máquina ao longo do tempo, é possível detectar alterações de *hardware* quer ao nível da substituição quer ao nível da adição de componentes. Estas alterações são claramente visíveis na opção de histórico quer para máquinas quer para componentes de *hardware* pois fornece ao técnico uma perspectiva do caminho percorrido por uma dada máquina ou por um componente específico.

Efectivamente, a identificação das máquinas é feita recorrendo a um algoritmo que, através dos códigos de identificação dos principais componentes de um PC, calcula um identificador da máquina que a tornará única, bem como os componentes que a constituem.

Uma outra funcionalidade associada à gestão das máquinas é a possibilidade de criação de grupos de máquinas. Com esta funcionalidade é possível, por exemplo, criar grupos específicos para locais remotos ou até mesmo grupos departamentais. O administrador poderá depois delegar a gestão de um ou mais grupos a técnicos (secção **5.2**).

5.5 Monitorização Online

A monitorização *online*, disponível no menu em `Gestão onLine`, corresponde ao acto de gestão em tempo real ou seja, o administrador/técnico pode consultar a informação desejada sobre as máquinas do universo gerido e actuar sobre elas em tempo real. A monitorização *online* encontra-se dividida em várias subcategorias, cada uma para uma área em particular. As subcategorias disponíveis são apresentadas na figura 54.

Figura 54 – Subcategorias de gestão para a gestão em tempo real (online)

Nas subsecções que se seguem será feita uma breve descrição da informação disponibilizada em cada uma das subcategorias bem como das tarefas que podem ser realizadas em tempo real sobre as máquinas.

5.5.1 Sistema Operativo

A consulta de informação relativa ao **sistema operativo** inclui a recolha de alguns dados de interesse bem como algumas tarefas que podem ser executadas remotamente sobre as máquinas conforme a figura 55.

Figura 55 – Consulta online de dados do sistema operativo

Toda a informação que caracteriza o sistema operativo da máquina alvo provém, em tempo real, dessa mesma máquina. Os técnicos podem consultar o nome do sistema operativo, o registo do mesmo, o *uptime* da máquina, a data de instalação do sistema operativo, o último *Service Pack* instalado, o tipo de sistema (e.g. *workstation* ou *server*), o número de série do sistema operativo e a data/hora local na máquina. Quanto às tarefas que os técnicos podem executar são a actualização da data/hora da máquina e a actualização da *product key* do sistema operativo (vulgarmente designada por licença).

5.5.2 Computador

Quanto ao **computador** ou máquina, é possível obter o tipo de sistema, o nome, a arquitectura, fabricante e modelo, bem como o papel desempenhado e o domínio de rede em que se encontra (figura 56).

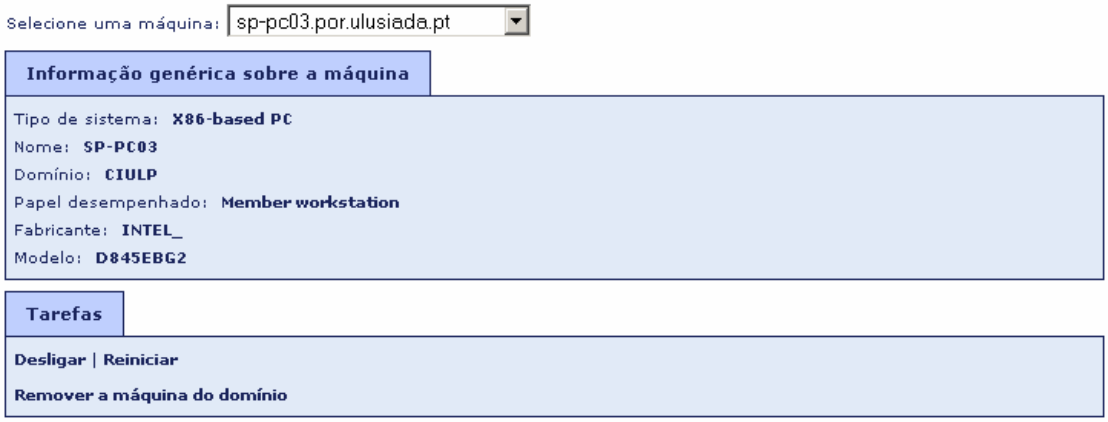


Figura 56 – Consulta online de dados referentes ao computador

As operações que podem ser realizadas sobre as máquinas incluem desligar ou reiniciar a máquina e removê-la do domínio em que se encontra. Note-se que as máquinas apenas podem ser geridas via WMI caso pertençam ao mesmo domínio de rede do sistema gestor. Logo, a remoção da máquina do domínio de rede em que se encontra eliminará automaticamente a máquina do universo de máquinas geridas.

5.5.3 Hardware

Foi já visto na secção **5.4** que uma máquina do universo gerido é constituída por um conjunto de componentes, sendo essa informação actualizada periodicamente de modo a actualizar o inventário do parque gerido bem como detectar eventuais situações anómalas ou substituição de componentes. Nesta secção de monitorização online, não se pretende que a consulta dos dados relativos ao *hardware* das máquinas seja exaustiva mas sim que disponibilize aos técnicos uma leitura resumida da constituição da máquina em termos dos

seus principais componentes. A figura 57 ilustra o levantamento de *hardware* efectuado para uma máquina.

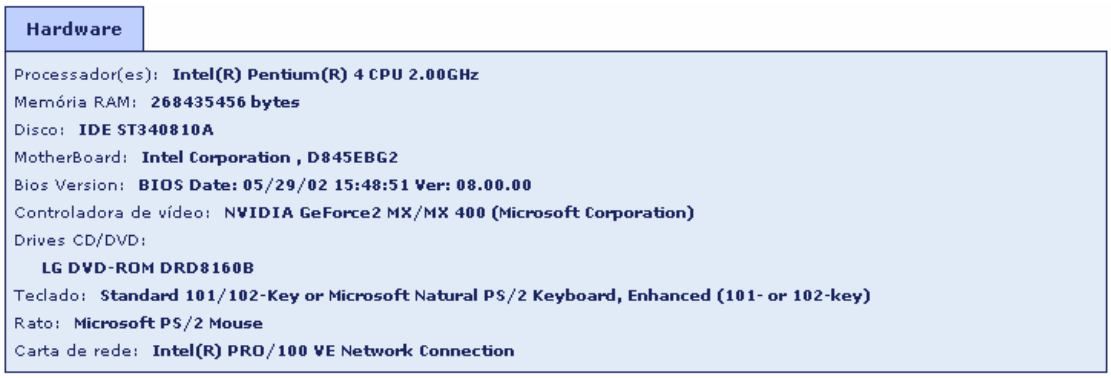


Figura 57 – Consulta online do hardware de uma máquina

A informação disponibilizada contempla o processador, o volume de memória física, a identificação do(s) disco(s) rígido(s), a identificação da *motherboard*, a versão da BIOS, a controladora de vídeo (vulgo placa gráfica), os leitores de CD/DVD, o tipo de teclado, o tipo de rato e a identificação da carta de rede. Para obter informação mais detalhada sobre o *hardware* da máquina, o técnico deverá aceder ao módulo de gestão de máquinas e seleccionar a máquina desejada na opção `Listagem` (ver figura 51). Aí terá acesso a informação detalhada sobre os vários componentes.

5.5.4 Memória

Relativamente à utilização dos recursos de memória, a monitorização *online* permite consultar a memória instalada, a memória livre e a memória virtual disponível no instante da consulta (figura 58). Note-se que a informação disponibilizada nesta subcategoria corresponde a uma espécie de "fotografia instantânea" pelo que não é possível aferir da carga da máquina para um período de tempo maior.

Figura 58 – Consulta online do estado da memória

5.5.5 Utilizadores

No que se refere a utilizadores, é um pouco escassa a informação disponibilizada pelo WMI. Os únicos dados de real interesse que podem ser disponibilizados são o utilizador

actual da máquina (aquele que se encontra na máquina na altura da consulta) e a duração da sessão desse mesmo utilizador tal como ilustra a figura 59.

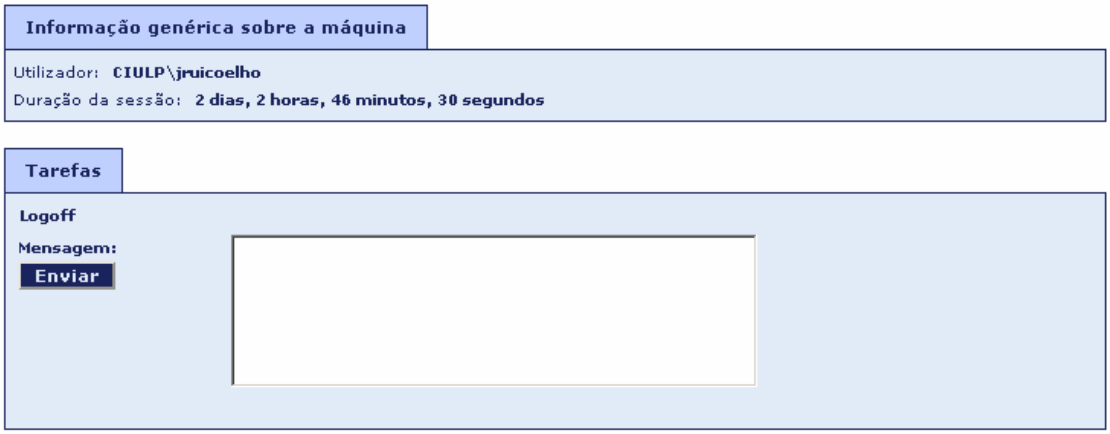


Figura 59 – Consulta online da sessão actual

Quanto às tarefas disponibilizadas ao técnico incluem o *logoff* remoto do utilizador que se encontra com a sessão activa na máquina e o envio de uma mensagem para o mesmo. O envio das mensagens é feito recorrendo ao serviço `Net Send` do *Windows*.

O facto de o modelo de informação disponibilizar apenas esta informação relativa aos utilizadores da máquina acaba por ser uma grande limitação já que nem o esquema CIM nem as extensões Win32 disponibilizam outros dados de elevado interesse como o histórico de utilizadores na máquina. Teria, obviamente, todo o interesse, o técnico poder saber quem esteve a utilizar a máquina num dado período. No entanto, esta limitação pode ser ultrapassada recorrendo a artifícios externos como por exemplo o uso de uma *script* que registe a autenticação do utilizador numa base de dados.

5.5.6 Processos

Apesar de ser um tanto ou quanto intrusivo (do ponto de vista ético), a listagem de processos pode ser uma ferramenta bastante útil para, por exemplo, detectar *software* impróprio a ser executado ou programas que estão a ser carregados na fase de *startup* do sistema operativo sem que tal seja necessário. Com esta funcionalidade, o administrador pode consultar em tempo real os processos que estão a ser executados na máquina e detectar, por exemplo, *spyware* ou *adware*. Consequentemente, e dada a interactividade proporcionada pela plataforma, o administrador pode, rapidamente, terminar esse(s) processo(s) e averiguar o motivo pelo qual o processo se encontrava em execução. Note-se, contudo, que com esta funcionalidade não se pretende criar um detector de *adware*, *spyware* ou qualquer outro tipo de *software* intrusivo. Pretende-se apenas verificar processos em execução ou no *startup* e

avaliar a utilização de recursos por parte dos mesmos como por exemplo, a memória utilizada por cada processo.

A figura 60 representa a lista de processos em execução para um determinado instante numa máquina. Para cada um dos processos constantes da lista, é possível terminá-lo. Esta operação pode ser realizada quer para libertação de recursos quer para controlo do *software* em execução no parque informático.

Selecione uma máquina: sp-pc03.por.ulusiada.pt Tipo de processos: Em execução

Processo	Iniciado	Path	Memória (Kb)	Opções
System Idle Process			16	
System			41	
smss.exe	15-12-2005 8:04:04	C:\WINDOWS\System32\smss.exe	45	
csrss.exe	15-12-2005 8:04:06	C:\WINDOWS\system32\csrss.exe	1585	
winlogon.exe	15-12-2005 8:04:06	C:\WINDOWS\system32\winlogon.exe	3117	
services.exe	15-12-2005 8:04:07	C:\WINDOWS\system32\services.exe	1692	
lsass.exe	15-12-2005 8:04:07	C:\WINDOWS\system32\lsass.exe	2298	
svchost.exe	15-12-2005 8:04:08	C:\WINDOWS\system32\svchost.exe	2331	
svchost.exe	15-12-2005 8:04:08	C:\WINDOWS\system32\svchost.exe	1843	
svchost.exe	15-12-2005 8:04:08	C:\WINDOWS\System32\svchost.exe	10973	
svchost.exe	15-12-2005 8:04:08	C:\WINDOWS\System32\svchost.exe	733	
svchost.exe	15-12-2005 8:04:09	C:\WINDOWS\System32\svchost.exe	791	
spoolsv.exe	15-12-2005 8:04:11	C:\WINDOWS\system32\spoolsv.exe	1798	
ccSetMgr.exe	15-12-2005 8:04:12	C:\Program Files\Common Files\Symantec Shared\ccSetMgr.exe	2015	
DefWatch.exe	15-12-2005 8:04:12	C:\Program Files\Symantec AntiVirus\DefWatch.exe	156	
MDM.EXE	15-12-2005 8:04:12	C:\Program Files\Common Files\Microsoft Shared\VS7Debug\mdm.exe	459	
ngctw32.exe	15-12-2005 8:04:12	C:\Program Files\Symantec\Ghost\ngctw32.exe	1942	
pctspk.exe	15-12-2005 8:04:13	C:\WINDOWS\system32\pctspk.exe	250	
SavRoam.exe	15-12-2005 8:04:14	C:\Program Files\Symantec AntiVirus\SavRoam.exe	930	
svchost.exe	15-12-2005 8:04:14	C:\WINDOWS\System32\svchost.exe	1929	
Rtvscan.exe	15-12-2005 8:04:14	C:\Program Files\Symantec AntiVirus\Rtvscan.exe	3138	
ccEvtMgr.exe	15-12-2005 8:04:15	C:\Program Files\Common Files\Symantec Shared\ccEvtMgr.exe	1606	
alg.exe	15-12-2005 8:04:21	C:\WINDOWS\System32\alg.exe	258	
explorer.exe	15-12-2005 9:37:18	C:\WINDOWS\Explorer.EXE	9445	
ccApp.exe	15-12-2005 9:37:26	C:\Program Files\Common Files\Symantec Shared\ccApp.exe	4776	
VPTray.exe	15-12-2005 9:37:26	C:\PROGRA~1\SYMANT~1\VPTray.exe	3142	
msmsgs.exe	15-12-2005 9:37:29	C:\Program Files\Messenger\MSMSGS.EXE	897	
LUCOMS~1.EXE	15-12-2005 9:37:53	C:\PROGRA~1\Symantec\LIVEUP~1\LUCOMS~1.EXE	3084	
acad.exe	15-12-2005 9:38:01	C:\Program Files\AutoCAD 2004\acad.exe	23097	
~e5d141.tmp	15-12-2005 9:38:02	C:\DOCUME~1\21525801\LOCALS~1\Temp\~e5d141.tmp	868	
WSCommCntr1.exe	15-12-2005 9:39:00	C:\Program Files\Common Files\Autodesk Shared\WSCommCntr1.exe	2855	
wmiprvse.exe	15-12-2005 11:56:52	C:\WINDOWS\System32\wbem\wmiprvse.exe	5501	

Figura 60 – Listagem de processos em execução

Para cada processo, a listagem contempla a data/hora em que foi iniciado, a *path* (caminho) do executável que lhe deu origem, o volume de memória que consome e uma opção para terminar.

Além dos processos em execução é também possível consultar a lista de processos no *startup* ou seja, aqueles que arrancam automaticamente quando a o sistema operativo é iniciado. Um exemplo deste tipo de listagem encontra-se na página seguinte na figura 61.

Selecione uma máquina: z4-pc13.por.ulusiada.pt Tipo de processos: No startup

Descrição	Comando	Localização
desktop	desktop.ini	Startup
desktop	desktop.ini	Startup
MSMSGS	"C:\Program Files\Messenger\MSMSGS.EXE" /background	HKU\S-1-5-21-2470483688-3119984381-3454996187-25652\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
desktop	desktop.ini	Startup
desktop	desktop.ini	Common Startup
NGClient	C:\Program Files\Symantec\Ghost\ngctw32.exe	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
ccApp	"C:\Program Files\Common Files\Symantec Shared\ccApp.exe"	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
vptray	C:\PROGRA~1\SYMANT~1\VPTray.exe	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

Figura 61 – Lista de processos no *startup* do sistema operativo

Para este tipo de processos, além da descrição, consta da listagem o comando para executar o processo e a localização do mesmo. Aqui, a localização não corresponde propriamente a um *path* mas sim ao local a partir de onde o processo é lançado. Por exemplo, em sistemas operativos *Windows*, é extremamente comum encontrarmos processos de *startup* a serem lançados pelo *Registry*.

5.5.7 Serviços

A listagem de serviços permite consultar o estado da máquina em termos dos seus serviços e a respectiva configuração. Para cada um deles é possível visualizar o nome, a *path* a partir de onde foi lançado e o seu estado conforme a figura 62.

Serviço	Path	Estado	Opções
Alerter	C:\WINDOWS\System32\svchost.exe -k LocalService	Stopped	
Application Layer Gateway Service	C:\WINDOWS\System32\alg.exe	Running	
Application Management	C:\WINDOWS\system32\svchost.exe -k netsvcs	Stopped	
ASP.NET State Service	C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322\aspnet_state.exe	Stopped	
Windows Audio	C:\WINDOWS\System32\svchost.exe -k netsvcs	Running	
AutoComplete Service	C:\Program Files\Internet Sweeper Pro\autocomp.exe	Stopped	
Background Intelligent Transfer Service	C:\WINDOWS\System32\svchost.exe -k netsvcs	Running	
Computer Browser	C:\WINDOWS\System32\svchost.exe -k netsvcs	Running	
...	...	...	
Universal Plug and Play Device Host	C:\WINDOWS\System32\svchost.exe -k LocalService	Stopped	
Windows Time	C:\WINDOWS\System32\svchost.exe -k netsvcs	Running	
WebClient	C:\WINDOWS\System32\svchost.exe -k LocalService	Running	
Windows Management Instrumentation	C:\WINDOWS\system32\svchost.exe -k netsvcs	Running	
Portable Media Serial Number Service	C:\WINDOWS\System32\svchost.exe -k netsvcs	Stopped	
Windows Management Instrumentation Driver Extensions	C:\WINDOWS\System32\svchost.exe -k netsvcs	Stopped	
Wireless Zero Configuration	C:\WINDOWS\System32\svchost.exe -k netsvcs	Running	
Network Provisioning Service	C:\WINDOWS\System32\svchost.exe -k netsvcs	Stopped	
Total: **96** serviços.			

Figura 62 – Listagem de serviços

A lista dispõe ainda de um mecanismo de filtragem de acordo com o estado do serviço: os que estão em execução, os que foram parados e os que estão em pausa. A selecção dos serviços para consulta por estado está apresentada na figura 63.

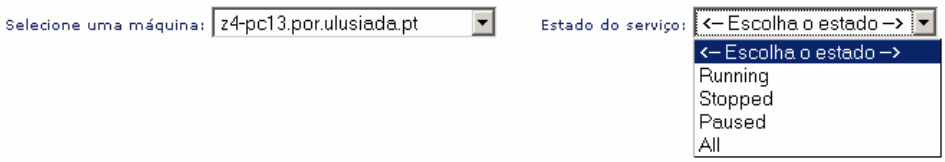


Figura 63 – Selecção de serviços para consulta de acordo com o seu estado

Para qualquer um dos serviços, o técnico tem acesso a um conjunto de informações detalhadas sobre o serviço bem como a capacidade de alterar o estado do serviço (`Run`, `Stop`, `Resume` e `Pause`) bem como modificar o modo de arranque do mesmo (`Boot`, `System`, `Auto`, `Manual` e `Disabled`). O acesso a esta área é feito através da coluna "`Opções`" na listagem da figura 63. A figura 64 apresenta a informação completa acerca do serviço `Logical Disk Manager`.

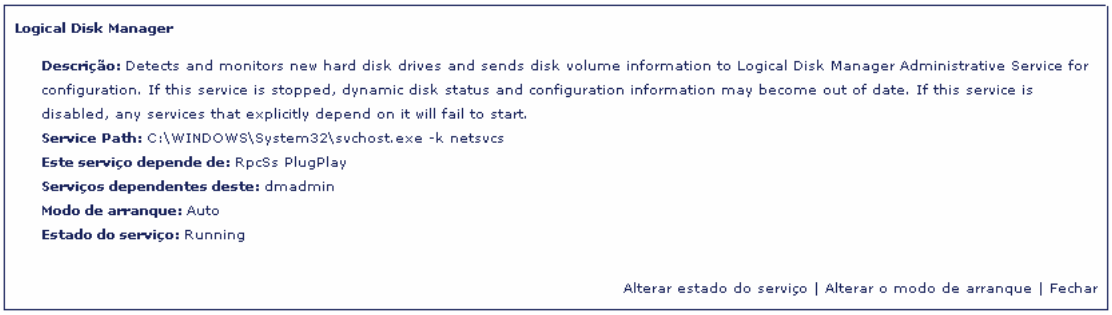


Figura 64 – Informação detalhada sobre um serviço

À semelhança da listagem de processos, também para os serviços o técnico pode alterar os seus estados, terminando-os ou iniciando-os ou até colocando-os no estado de pausa. Em alguns casos, alterações feitas ao estado do serviço (e.g. alteração do estado de STOPPED para RUNNING) podem depender do modo de arranque configurado (e.g. não é possível iniciar um serviço cujo modo de arranque esteja em *DISABLE*). A figura 65 ilustra o modo como esta alteração pode ser efectuada.

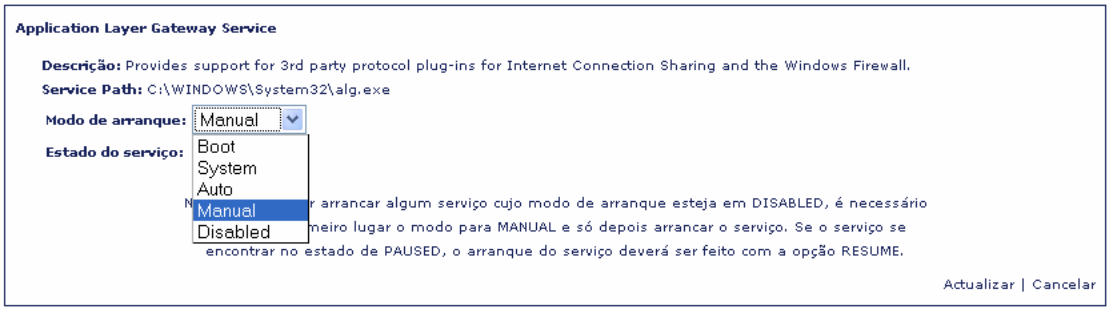


Figura 65 – Alteração do modo de arranque de um serviço

Na figura anterior é apresentada a interface de alteração do modo de arranque do serviço `Application Layer Gateway Service`. A alteração será executada em tempo real na máquina assim que o técnico "`Actualizar`" o serviço (opção no canto inferior direito).

5.5.8 Configuração de Rede

No que respeita à configuração de rede, os técnicos podem consultar quer a configuração do adaptador de rede quer a tabela de roteamento da máquina. A informação disponibilizada para o adaptador de rede está apresentada na figura 66.

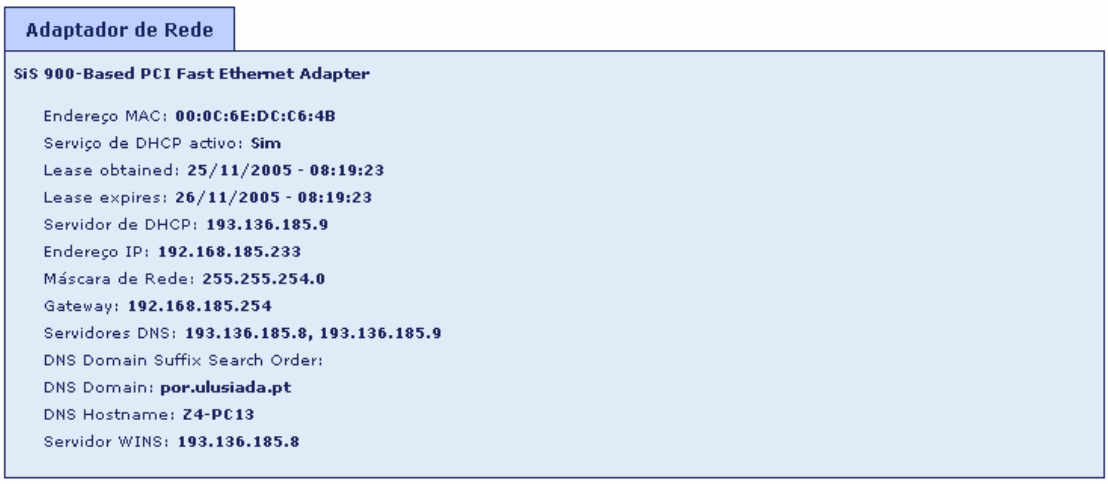


Figura 66 – Configuração do adaptador de rede

A configuração do adaptador de rede contempla o endereço MAC (endereço físico da carta de rede), informação acerca da *lease* DHCP (caso o serviço esteja activo), o endereço IP, a máscara de rede, a *gateway*, a lista de servidores de DNS, os sufixos de domínio DNS para pesquisa, o *hostname* da máquina e por fim o servidor de WINS. Quanto à tabela de roteamento, encontra-se na figura 67.

Tabela de Roteamento

Destino	Máscara	Gateway	Interface	Métrica
0.0.0.0	0.0.0.0	192.168.185.254	192.168.185.151	20
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
192.168.184.0	255.255.254.0	192.168.185.151	192.168.185.151	20
192.168.185.151	255.255.255.255	127.0.0.1	127.0.0.1	20
192.168.185.255	255.255.255.255	192.168.185.151	192.168.185.151	20
224.0.0.0	240.0.0.0	192.168.185.151	192.168.185.151	20
255.255.255.255	255.255.255.255	192.168.185.151	192.168.185.151	1

Figura 67 – Tabela de roteamento

Além da consulta da configuração e tabela de roteamento associada ao adaptador, o administrador pode actuar directamente sobre a máquina alterando parâmetros como servidores de DNS, servidores de WINS ou a *gateway*. Pode, inclusive, solicitar a renovação

da licença de DHCP apenas nos casos em que as máquinas não tenham endereço IP fixo. Estas operações estão ilustradas na figura 68.

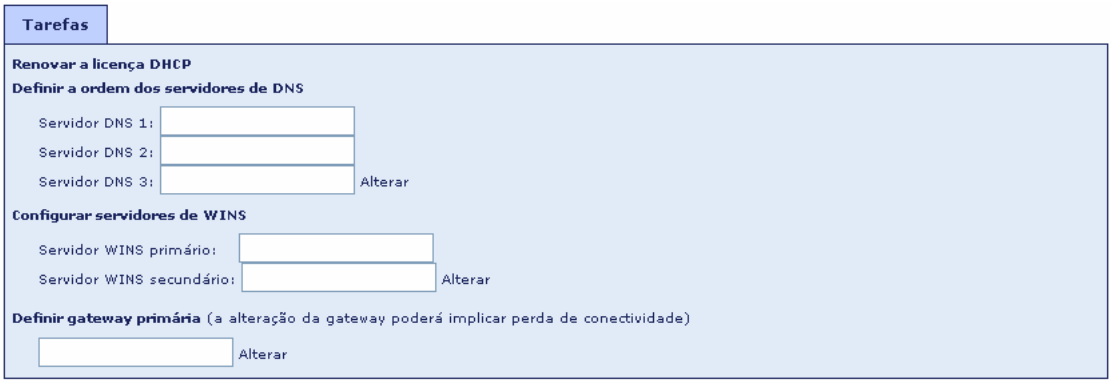


Figura 68 – Tarefas para alteração das configurações de rede

5.5.9 Conexões de Rede

A listagem de conexões de rede permite obter uma fotografia instantânea do estado das ligações TCP/IP da máquina alvo. Note-se, contudo, que a informação apresentada é efémera dado que diz respeito a um instante temporal. Esta informação é obtida via SNMP o que requer que o serviço esteja a correr na máquina alvo. No caso de o serviço não estar a correr, o administrador pode, recorrendo à interface dos serviços (**5.5.7**), activá-lo, e assim obter a listagem. A figura 69 apresenta um exemplo.

Endereço Local	Porto Local	Endereço Remoto	Porto Remoto	Estado	Opções
0.0.0.0	135	0.0.0.0	2192	listen	
0.0.0.0	445	0.0.0.0	30936	listen	
127.0.0.1	12583	127.0.0.1	6000	established	
127.0.0.1	12593	127.0.0.1	12594	established	
127.0.0.1	12594	127.0.0.1	12593	established	
193.136.185.25	80	193.136.185.25	13028	established	
193.136.185.25	139	0.0.0.0	14518	listen	
•••	•••	•••	•••	•••	•••
193.136.185.25	4662	219.89.234.14	2605	established	
193.136.185.25	12726	195.245.244.243	4661	established	
193.136.185.25	12972	87.103.59.247	17389	established	
193.136.185.25	13018	80.167.190.27	4662	established	
193.136.185.25	13027	193.136.3.198	80	established	
193.136.185.25	13028	193.136.185.25	80	established	
193.136.185.25	13029	193.136.185.21	3306	timeWait	
193.136.185.25	13030	193.136.185.25	135	timeWait	
Total: **32** conexões.					

Figura 69 – Listagem de conexões TCP/IP

Para cada uma das ligações apresentadas é possível terminá-la, fechando a ligação.

Na altura em que foi desenvolvida esta funcionalidade, e durante o decorrer deste trabalho, nem o modelo CIM nem as extensões Win32 disponibilizavam qualquer tipo de classe que permitisse listar as conexões activas na máquina alvo pelo que foi necessário

recorrer ao SNMP. A desvantagem desta solução é a limitação dos dados obtidos via SNMP: nada mais que um simples descrição da conexão e dos seus parâmetros.

Algo que seria extremamente interessante como complemento da listagem de conexões seria conhecer os processos ou aplicações responsáveis pelas mesmas. Do ponto de vista da administração das máquinas é preferível saber que aplicação ou processo está a iniciar uma conexão indesejada e eliminar a aplicação ou processo em vez de terminar a ligação aberta, correndo o risco de ela tornar a abrir.

5.5.10 Software Instalado

Quanto ao *software* instalado, o processo de recolha e apresentação da informação é semelhante à funcionalidade anterior, conforme ilustra a figura 70.

ID	Software	Tipo	Data de instalação
0	3Com NIC Diagnostics	Application	2003-11-14, 13:26:16
1	EA SPORTS online 2006	Application	2005-10-18, 15:50:8
2	Ad-aware 6 Personal	Application	2003-11-17, 17:7:10
3	Adobe Photoshop 7.0	Application	2004-8-25, 13:34:56
4	Adobe Premiere 6.5	Application	2004-7-23, 15:36:56
5	AI - Series	Application	2003-11-14, 18:19:6
6	ASUS Probe V2.20.07	Application	2003-11-14, 18:12:36
7	AsusUpdate	Application	2004-6-28, 18:46:0
8	Advanced Uninstaller PRO 2003 version 6	Application	2004-8-4, 12:14:16
9	Btuga Revolution 1.6.1	Application	2005-1-25, 9:52:54
10	CCE SP Trial Version	Application	2004-1-5, 18:44:12
11	DiamondCS TDS-3	Application	2004-10-29, 11:28:28
12	DVD Decrypter (Remove Only)	Application	2004-4-7, 17:5:52
13	DVD Shrink 3.2	Application	2005-12-9, 12:34:28
14	DVD2DVD-R 1.8.0 Beta 3	Application	2004-1-5, 19:3:58
15	eMule	Application	2004-8-4, 11:40:44
16	Ethereal 0.10.12	Application	2005-10-19, 10:54:26
17	F-Secure SSH Client	Application	2003-12-4, 10:32:26
18	FileZilla (remove only)	Application	2004-12-7, 12:57:46
19	Indeo® software	Application	2005-1-11, 13:0:58
20	CamStudio	Application	2004-7-21, 11:48:18
21	iReasoning MIB Browser (remove only)	Application	2005-9-6, 17:18:54
22	Java Device Manager	Application	2005-3-15, 12:45:54
23	Windows XP Hotfix - KB834707	Application	2004-10-14, 9:18:58
24	Windows XP Hotfix - KB867282	Application	2005-2-9, 10:49:52
25	Microsoft Data Access Components KB870669	Application	2004-7-3, 14:5:6
26	Windows XP Hotfix - KB873333	Application	2005-2-9, 10:50:54
27	Windows XP Hotfix - KB873339	Application	2004-12-15, 10:32:34
•••	•••	•••	•••

Figura 70 – Listagem de software instalado

Tal como a listagem das conexões de rede, a lista de *software* instalado é obtida também por SNMP, dependendo por isso do estado do serviço na máquina alvo. Esta funcionalidade sofre também das limitações do SNMP não permitindo, por exemplo, a capacidade de desinstalar *software*. Relativamente ao *software* instalado e à possibilidade de o desinstalar remotamente, é possível fazê-lo através do WMI. No entanto, a listagem fornecida pelo WMI não permite obter uma panorâmica geral do *software* instalado dado que a listagem é bastante incompleta, não por limitação do modelo CIM mas pela sua implementação, neste

caso, por parte da Microsoft. Por este motivo, a escolha técnica recaiu sobre a utilização do SNMP, sacrificando a possibilidade de efectuar algumas desinstalações remotas.

5.6 Gestão de Tarefas Automáticas

O sistema de gestão de tarefas automáticas constitui a base de suporte à monitorização *offline*. A elaboração de relatórios, consulta de componentes de *hardware*, consulta de alterações nas configurações de rede, etc., só pode ser feita se o sistema de informação que suporta a plataforma de gestão contiver a referida informação. Para tal, o desenvolvimento da funcionalidade de gestão de tarefas automáticas permite aos técnicos configurar tarefas para recolha de dados, populando assim o sistema de informação com os dados necessários para posteriores consultas.

O acesso ao módulo de gestão de tarefas é feito pelo menu seleccionando a opção "`Tarefas`", conforme a figura 71.

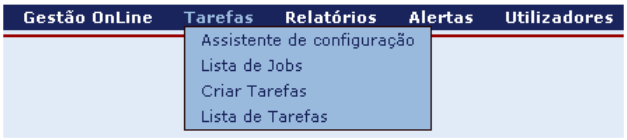


Figura 71 – Acesso ao módulo de gestão de tarefas

Antes de prosseguir na descrição das funcionalidades deste módulo, importa referir a diferença entre um *Job* e uma tarefa. Uma tarefa, tal como o próprio nome indica, consiste numa operação de recolha de dados de uma ou mais máquinas. A tarefa pode consistir na recolha de informação de estado, levantamento de *hardware*, configurações, etc. Um *Job*, é uma operação automática que executa uma tarefa de acordo com o período de execução configurado. Por exemplo, a leitura da configuração de rede de uma máquina pode-se constituir como uma tarefa. De modo a fazê-lo regularmente sem a necessidade de intervenção dos técnicos, é necessário criar um *Job*, com a periodicidade pretendida (uma vez por dia, uma vez por semana, etc.) e seleccionar a tarefa de leitura da configuração de rede para ser executada pelo *Job*.

A criação de tarefas é feita através da página respectiva e encontra-se dividida nas seguintes categorias: sistema operativo, computador, *hardware*/componentes, *networking*, *software* instalado, processos, serviços, avançada. Em cada uma das categorias, é possível criar tarefas para a recolha de vários dados a partir de uma ou mais máquinas. A título de exemplo, a figura 72 representa a interface para a criação de uma tarefa para o levantamento do *software* instalado nas máquinas.

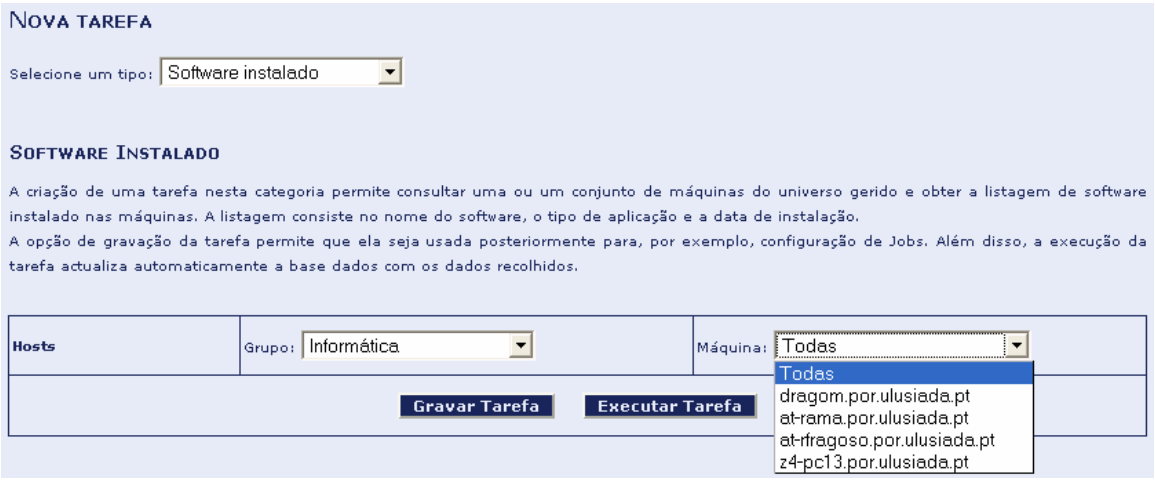


Figura 72 – Tarefa para leitura de *software* instalado

Na figura apresentada, trata-se da criação de uma tarefa para a leitura do *software* instalado em todas as máquinas do grupo "`Informática`". O técnico dispõe de dois botões: um para gravar a tarefa e outro para executar a tarefa. A execução da tarefa consiste numa consulta do *software* instalado nas máquinas seleccionadas em tempo real e actualização da base de dados com a informação recolhida. A gravação da tarefa permite a sua reutilização posteriormente quer em execuções manuais (invocadas pelo técnico) quer em execuções automáticas em *Jobs*.

Relativamente aos *Jobs*, foi desenvolvido um assistente de configuração (*wizard*), dividido em quatro passos e que permite ao técnico configurar todos os aspectos do novo *Job*.

Em primeiro lugar, o técnico deverá escolher o alvo do *Job* ou seja, sobre que máquina ou grupo de máquinas deverá incidir a tarefa (figura 73).

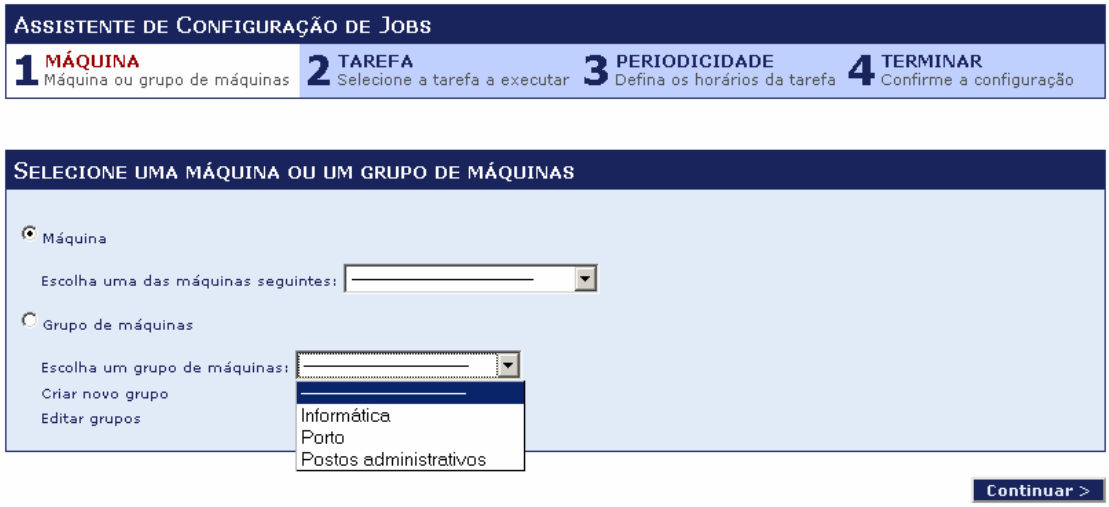


Figura 73 – Selecção da(s) máquinas na configuração de tarefas automáticas

Em seguida, deverá seleccionar a tarefa a executar sobre a(s) máquina(s) escolhida(s) conforme a figura 74. Aqui poderá escolher uma de entre várias categorias: sistema operativo,

computador, *hardware*/componentes, *networking*, *software* instalado, processos, serviços, avançada.

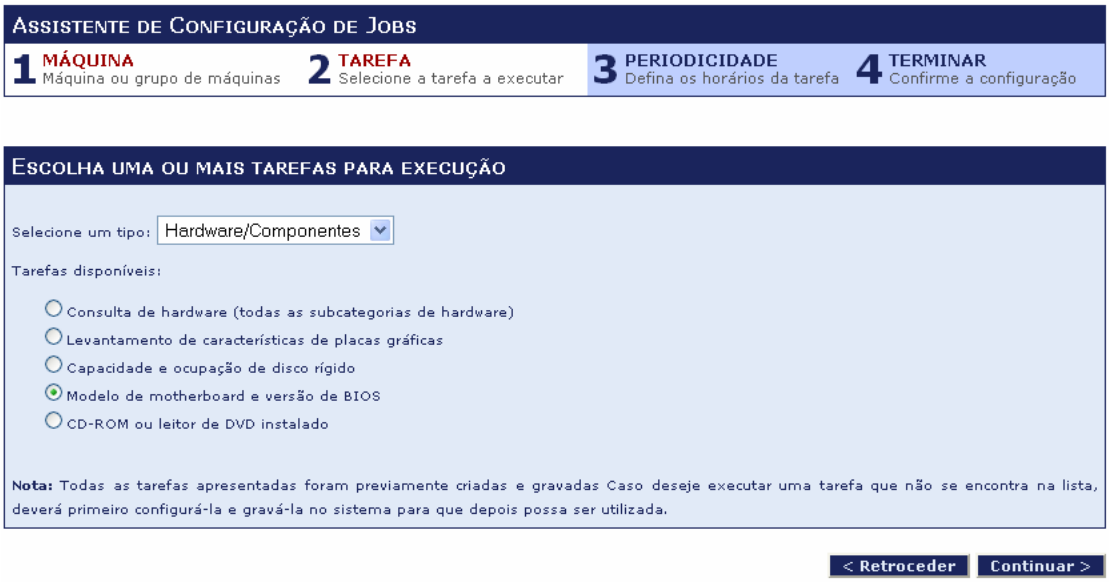


Figura 74 – Selecção da tarefa a executar para criação de *Jobs*

O passo seguinte consiste na definição da periodicidade de execução do *Job*. Dada a variedade de dados passíveis de recolha, a periodicidade da tarefa pode ser devidamente escolhida para satisfazer os requisitos da tarefa. Por exemplo, a monitorização do espaço em disco ocupado de um servidor deve ser controlada com uma periodicidade mais curta que o levantamento de números de série do *hardware* de um posto de trabalho. Os tipos de periodicidade disponíveis são os seguintes:

– **Apenas uma vez**: o técnico indica uma data e uma hora e o *Job* é executado apenas uma vez.

– **Hora**: permite configurar *Jobs* com periodicidade horária ou seja, podendo ser configurada uma hora de início e uma hora de fim, a taxa de repetição por hora e os dias da semana em que o *Job* será executado. A hora de início e a hora de fim servem apenas para evitar situações em que não é necessário, por exemplo, a recolha de dados durante a noite.

– **Dia**: criação de *Jobs* com periodicidade diária ou seja, o técnico indica a hora e os dias da semana em que deseja que o *Job* seja executado.

– **Semana**: a periodicidade semanal permite ao técnico indicar a hora e o dia para a execução do *Job*.

– **Mês**: criação de *Jobs* com periodicidade mensal. O técnico pode indicar o dia do mês ou o tipo de dia (ex. primeira segunda ou última sexta) bem como os meses em que deseja executar o *Job*.

A figura 75 na página seguinte ilustra a interface para definição de periodicidade do *Job*.

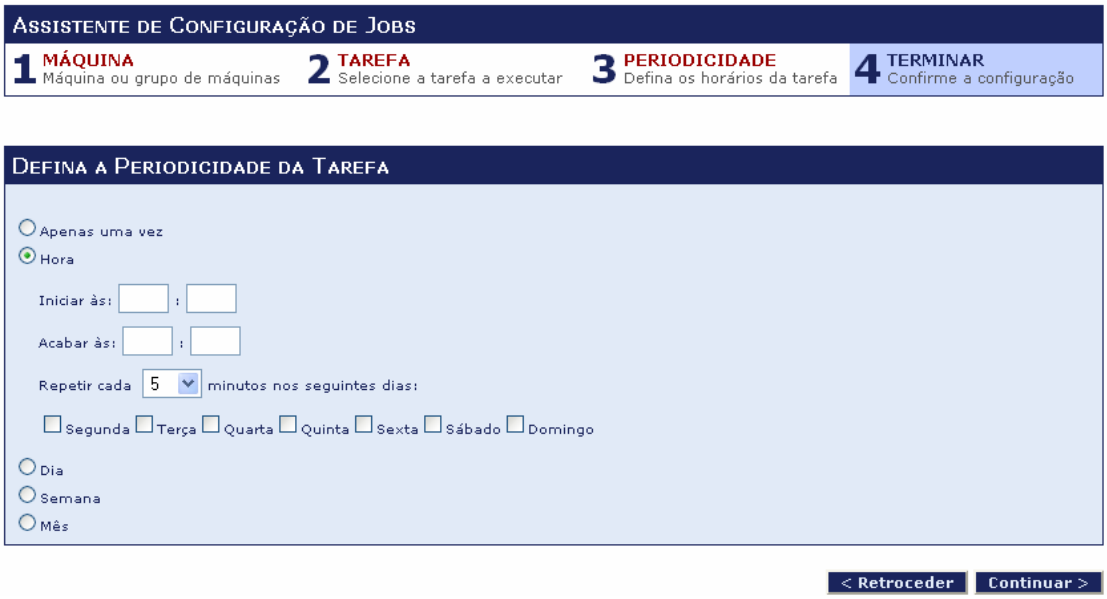


Figura 75 – Definição da periodicidade do *Job*

O último passo consiste na confirmação da tarefa automática a criar. Além da informação sobre os dados configurados nos pontos anteriores, como a(s) máquina(s) alvo, a tarefa e a sua periodicidade, é ainda necessário indicar uma data e hora a partir da qual o *job* se tornará activo. A figura 76 apresenta o resumo dos passos anteriores.

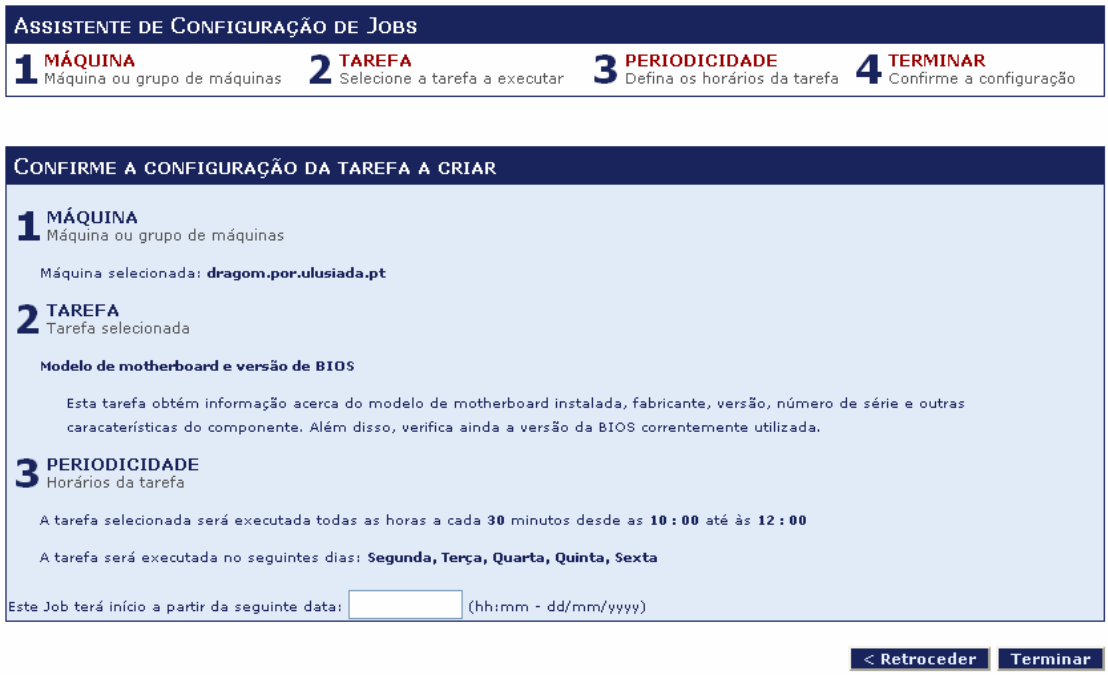


Figura 76 – Confirmação da tarefa automática a criar

Após "`Terminar`", o *Job* é criado no sistema operativo da máquina onde corre a plataforma de gestão e é também adicionado ao sistema de informação, juntamente com todos os dados específicos da sua configuração. Este registo em base de dados permite aos técnicos manterem um conhecimento actualizado dos *Jobs* que se encontram configurados.

5.7 Monitorização Offline

A monitorização *offline* é a funcionalidade que permite ao administrador consultar o estado do parque informático bem como aferir de mudanças na composição do parque e/ou composição das máquinas geridas. A monitorização *offline* permite ainda ao administrador o acesso a históricos de actividade quer em termos de operações de gestão efectuadas sobre as máquinas quer em termos de alterações de configuração nas próprias máquinas. Outra funcionalidade disponibilizada ao administrador é a capacidade de efectuar relatórios de acordo com critérios definidos pelo mesmo como por exemplo, relatório de actualizações para máquinas com um determinado sistema operativo.

Dada a natureza da mesma, a obtenção e listagem de históricos das máquinas está disponível juntamente com as restantes opções para a gestão de máquinas (ver figura 51). Esta opção de gestão permite aos técnicos conhecer o passado das máquinas geridas bem como as suas constituições em termos de componentes de *hardware*. É também possível obter históricos para os componentes e conhecer assim os "percursos" destes ao longo do tempo e, eventualmente, as várias máquinas onde estiveram instalados.

A funcionalidade de elaboração de relatórios é disponibilizada através da respectiva opção na barra de menu conforme a figura 77.

Figura 77 – Acesso ao módulo de monitorização *offline*

Um dos relatórios possíveis de gerar é o de *software* instalado. É possível, desde que hajam tarefas automáticas capazes de efectuar a recolha da informação periodicamente, gerar relatórios de *software* instalado no parque informático. A figura 78 na página seguinte ilustra a interface apresentada ao técnico para a geração do relatório.

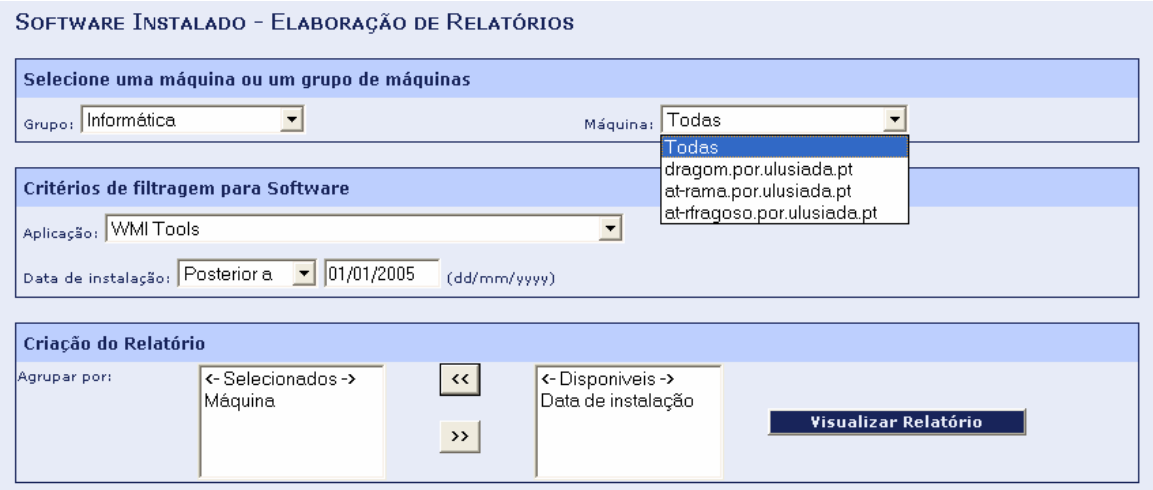


Figura 78 – Criação de relatórios de *software* instalado

A elaboração de um relatório é composta por vários passos que permitem refinar os dados apresentados, implementando alguns filtros. O exemplo apresentado na figura 78 define a configuração de um relatório elaborado para todas as máquinas do grupo "`Informática`" que têm instalada a aplicação "`WMI Tools`", com data de instalação superior a "`01/01/2005`". Os resultados da pesquisa serão agrupados por "`Máquina`". A figura 79 apresenta o relatório resultante dos critérios escolhidos.

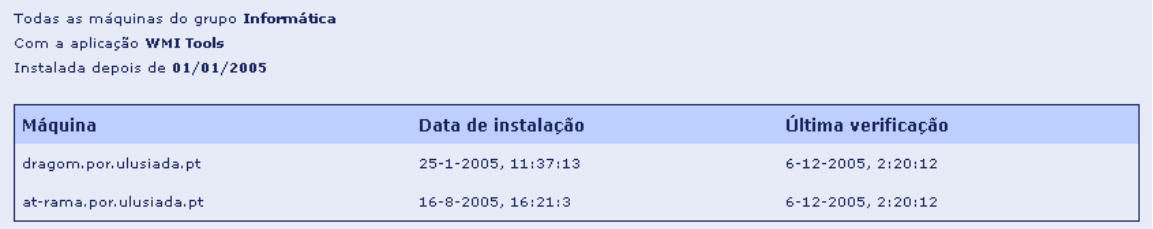


Figura 79 – Relatório de *software* instalado

A informação disponibilizada pela monitorização *offline* torna-se útil apenas quando existe ou seja, para que o administrador possa consultar o histórico das máquinas ou mesmo elaborar relatórios é necessário que exista informação em base de dados. Significa isto que é necessário que o sistema de gestão efectue recolhas periódicas de dados, tarefa esta que será da responsabilidade do próprio administrador e que deverá ser realizada recorrendo à configuração de tarefas automáticas - *Jobs* (ver secção **5.6**).

5.8 Alertas e Notificações

O sistema de alertas e notificações é o que representa, na prática, o carácter pró-activo da plataforma de gestão. De acordo com o que foi dito em **2.2.2**, a simples consulta do estado das máquinas como a listagem de processos ou conexões de rede bem como a verificação esporádica do *hardware* permite ao administrador um comportamento reactivo ou seja, no

caso de eventuais falhas ou problemas em alguma das máquinas do universo gerido, o administrador poderá apenas actuar *à posteriori*. A pró-actividade do sistema consiste precisamente na sua capacidade de detecção de eventuais problemas, alertando o(s) administrador(es) e possibilitando-lhe(s) a tomada de acções e/ou decisões que permitam, antecipadamente, eliminar a ocorrência do problema ou eventualmente, minorar as consequências devidas à ocorrência do mesmo. Por exemplo, para um determinado conjunto de máquinas com a mesma versão de sistema operativo, se eventualmente houver uma que não esteja actualizada, quer por versões de *drivers*, quer por falta de actualizações do próprio sistema operativo, o administrador terá todo o interesse em ser alertado para o facto, em vez de receber uma mensagem de quebra de funcionamento derivada a uma falha consequente da falta de actualização.

Ao nível da plataforma de gestão desenvolvida, o sistema de gestão de alertas e notificações foi simplificado, ou melhor, não tão elaborado quanto o resto das funcionalidades pois pretende-se apenas demonstrar a utilidade de um sistema deste género e as vantagens de uma gestão pró-activa. Além disso, o modelo CIM possui um modelo próprio para o tratamento de eventos (**3.3.1.1**) que pode e deve ser usado para o efeito. Este aspecto será analisado em maior detalhe no último capítulo deste trabalho, onde serão expostos alguns caminhos a seguir no futuro e que complementarão este trabalho.

Assim, e seguindo a mesma linha de organização quer da monitorização *online*, quer dos sistema de gestão de tarefas automáticas, foram criadas algumas subcategorias e um conjunto de alertas configuráveis por cada subcategoria. O alerta reside essencialmente no envio de um *mail* para o(s) administrador(es) configurados para o receber.

6 Conclusões

O trabalho desenvolvido no âmbito desta dissertação teve como principal objectivo avaliar a exequibilidade e o leque de vantagens que soluções centralizadas, baseadas em *standards*, podem proporcionar na gestão de um parque informático. Foi desenvolvida uma plataforma de gestão cujo núcleo assenta sobretudo na utilização do modelo de informação CIM e mostra a utilidade que um modelo de informação comum pode representar na gestão de parques informáticos. Trata-se de um sistema de gestão via *web*¸ o que permite que a sua utilização seja feita a partir de qualquer local. Além disso, toda a lógica do sistema de gestão reside na máquina onde este é instalado, não havendo a necessidade de instalação de *software* adicional nas máquinas a gerir.

Dado o carácter recente das tecnologias usadas, estas ainda não são implementadas pela maioria dos fabricantes pelo que foi necessário efectuar uma escolha na fase de projecto da plataforma. Dado que os sistemas operativos da Microsoft suportam nativamente o modelo CIM e a infra-estrutura WBEM através do WMI, optou-se por direccionar o desenvolvimento da plataforma para máquinas com sistemas operativos da Microsoft. Apesar deste estreitamento tecnológico forçado pela escolha realizada, a utilidade de uma plataforma de gestão deste género é, inequivocamente, inegável. Proporcionar ao administrador de sistemas a capacidade de gerir todo o seu parque informático, ainda que constituído por máquinas com sistemas operativos do mesmo fabricante, a partir de um único local é algo de extremamente poderoso e que transforma por completo a noção de gestão de um parque informático, centralizando-a e tornando-a eficiente, célere e, acima de tudo, pró-activa.

Exceptuando a escolha do WMI como única tecnologia proprietária, todas as restantes ferramentas utilizadas são *open-source* e as normas são *standards* não proprietários. Estão portanto disponíveis gratuitamente bem como o código fonte pode ser alterado em qualquer altura para satisfazer eventuais novos requisitos da plataforma. Além disso, a própria plataforma constitui-se como um projecto *open-source* podendo vir a ser disponibilizada para eventuais melhoramentos ou adição de funcionalidades. As linguagens de programação utilizadas foram o PHP e o Perl e o sistema de informação interno foi criado em MySQL. O *framework web* a partir do qual se iniciou o desenvolvimento é também um projecto *open-source* de seu nome PRADO.

A utilização do WMI justifica-se por várias razões. Em primeiro lugar, o carácter recente das tecnologias usadas fez com que se tornasse impossível caminhar para uma solução de gestão de parques heterogéneos. Os únicos sistemas operativos com suporte nativo da infra-estrutura WBEM são os da Microsoft e, no entanto, não o fazem completamente de acordo com as normas, utilizando a referida tecnologia proprietária, o WMI. A grande diferença na implementação da Microsoft reside no acesso aos dados de gestão, sendo feito por objectos COM e não pelo uso das normas "Operações CIM sobre HTTP" e "Representação de CIM em XML". Esta característica condicionou, logo à partida, o trabalho que se iria realizar. Contudo, para que esta escolha não condicionasse o projecto a médio prazo e dado que a plataforma segue uma estrutura modular, o acesso à informação das máquinas constitui um módulo estanque, a partir do qual não dependem as funcionalidades de gestão pelo que, no futuro, poderá ser facilmente alterado e adaptado às normas assim que estas sejam implementadas pelos vários fabricantes.

Quanto à plataforma de gestão desenvolvida, essencialmente, disponibiliza ao administrador três grandes blocos de gestão: monitorização/gestão *online*, monitorização *offline* e sistema de alertas e notificações.

A monitorização/gestão *online* permite ao administrador consultar em tempo real o estado das máquinas geridas de acordo com um conjunto de subcategorias. Desde a informação relativa ao sistema operativo até ao levantamento do *hardware* da máquina, passando por exemplo pelas configurações de rede, o administrador pode em qualquer altura, ter um conhecimento aprofundado do estado do parque informático. Além da consulta de dados, é também possível, em tempo real, alterar configurações ou executar operações remotamente sobre as máquinas geridas. Terminar processos em execução, alterar o estado dos serviços do sistema operativo ou o seu modo de arranque, definir servidores de DNS são apenas alguns exemplos das operações que o(s) administrador(es) pode executar remotamente sobre as máquinas em tempo real.

O módulo de monitorização *online* permite que o administrador tenha, em qualquer altura, acesso ao estado actual do parque informático. No entanto, a gestão de um parque informático envolve também o conhecimento de estados anteriores e o histórico das máquinas. Para este efeito, a plataforma disponibiliza um módulo de monitorização *offline* que, através de tarefas automáticas de recolha de dados e armazenamento dos mesmos no sistema de informação interno, permite ao administrador consultar históricos e gerar relatórios de acordo com um leque de categorias: *hardware*, *software* instalado, configuração de rede, serviços, etc. A monitorização *offline* torna-se útil quando existe informação suficiente em base de dados ou seja, quando existem tarefas automáticas de recolha de dados configuradas para a obtenção de informação específica sobre as máquinas. Por exemplo, a elaboração de um relatório sobre alterações de composição de uma máquina em termos do seu *hardware* só pode ser feita se

houver uma tarefa automática que, periodicamente, faça o levantamento dos componentes de *hardware* da máquina gerida e armazene essa informação na base de dados. Para a criação e configuração das tarefas automáticas foi desenvolvido um módulo de *frontend* que através de uma interface *web* permite que o administrador crie uma tarefa automática. Após criar a tarefa automática, a plataforma encarrega-se de integrar a tarefa no próprio sistema operativo transferindo a execução periódica da tarefa para o motor de *scheduling* do sistema operativo onde se encontra instalada a plataforma de gestão.

O sistema de alertas e notificações é o componente que confere à plataforma a capacidade de gestão pró-activa. A consulta em tempo real do estado das máquinas ou a análise de relatórios sobre os mais variados aspectos da gestão do parque informático representam a adopção de uma filosofia de gestão reactiva, ou seja, após a ocorrência de um problema o administrador servir-se-á das funcionalidades referidas para identificar eventuais causas ou métodos de resolução para o problema ocorrido. A disponibilização de um sistema de alertas e notificações fornece ao administrador a capacidade de detecção prévia de eventuais problemas que poderão ocorrer. Ao receber um alerta ou uma notificação sobre o estado anormal de uma máquina, o administrador pode tomar as devidas acções e/ou decisões que permitam evitar a ocorrência do problema ou, pelo menos, minorar as consequências da ocorrência do problema.

Durante a fase de desenvolvimento e implementação da plataforma, foram vários os problemas que foram surgindo. Um aspecto já referido prende-se com o carácter recente das tecnologias utilizadas pelo que o caminho a percorrer para uma gestão completa de parques heterogéneos é, ainda, longo.

Outro problema associado ao facto de a iniciativa WBEM ser recente tem a ver com a própria utilização do modelo CIM. O que se verificou na fase de implementação foi que a parte comum do modelo CIM (o núcleo e os modelos comuns) eram extremamente parcos na quantidade e qualidade de informação disponibilizada para consulta. Grande parte da informação necessária para gestão foi encontrada nas extensões do modelo, desenvolvidas pela própria Microsoft. Apesar da organização e estruturação da informação nas extensões respeitar as normas do modelo comum, esta será uma das grandes adversidades no futuro pois se todos os fabricantes desenvolverem extensões para a gestão dos seus equipamentos, será mais difícil e complexa a tarefa de um sistema de gestão centralizada.

A utilização do protocolo de gestão de redes SNMP no âmbito deste trabalho foi necessária dadas algumas limitações apresentadas pelas extensões da Microsoft ao modelo CIM. Estas limitações consistem na inexistência de qualquer tipo de estrutura ou informação capaz de fornecer o estado de uma máquina em termos das suas conexões de rede e na listagem incompleta de *software* instalado nas máquinas. Quanto à primeira, a sua inexistência fez com que desde logo se procurassem outras soluções para encontrar a

informação desejada. Aqui, o SNMP apresentou-se como uma ferramenta capaz pois disponibiliza a tabela de conexões de rede na máquina. As desvantagens residem no facto de, na máquina gerida, ser necessário que o serviço de SNMP esteja instalado e em execução e por outro lado, a incapacidade de saber qual o processo responsável pela ligação. Quanto à listagem de *software* instalado nas máquinas, as extensões da Microsoft fornecem uma listagem demasiado incompleta pelo que, também neste ponto, se recorreu ao SNMP em seu detrimento. A razão desta escolha deve-se ao facto de o SNMP disponibilizar uma listagem completa de todo o *software* instalado nas máquinas geridas. No entanto, a opção pelo SNMP invalida qualquer tipo de gestão avançada de *software* como por exemplo a capacidade de desinstalação remota do mesmo, funcionalidade disponibilizada pelas extensões da Microsoft ao modelo CIM.

Além do desenvolvimento da plataforma de gestão, esta dissertação procurou, essencialmente, trazer à luz do dia os esforços que tem vindo a ser desenvolvidos por várias entidades, dentre as quais se destaca claramente o DMTF e a utilidade, talvez mesmo necessidade, da gestão de parques informáticos baseada nestas tecnologias. Obviamente, o papel principal vai para o modelo de informação comum CIM, integrador dos subsistemas actuais de gestão disponibilizados pelos vários fabricantes. Dada a sua neutralidade em termos tecnológicos, o modelo CIM permite aceder, através de uma única interface normalizada, aos mais variados tópicos desde sistema operativo até ao *hardware*, permitindo actuar em cada um deles, executando operações ou alterando configurações.

6.1 Considerações Finais

Nos últimos anos, pelo menos desde o proliferar dos computadores pessoais e do crescimento exponencial dos parques informáticos, temos assistido a um desleixo praticamente a todos os níveis no que se refere à gestão dos parques informáticos. Desde a simples lacuna na gestão até situações em que um técnico de informática tem que se deslocar quilómetros para resolver um problema de configuração num computador pessoal que se encontra num escritório remoto, várias são as situações para as quais urge procurar soluções e implementá-las de facto, independentemente dos fabricantes, da constituição do *hardware* ou do tipo de sistema operativo usado nas máquinas.

Apesar de ainda verdes, os esforços desenvolvidos no sentido da uniformização da gestão de parques informáticos começam a dar os seus frutos e a mostrar as suas reais capacidades. Nos últimos dez anos temos assistido ao aparecimento de *standards* desenvolvidos para o efeito como é o caso do DMI, do CIM ou do WBEM.

Infelizmente, e devido sobretudo ao carácter recente do WBEM e das versões finais dos *standards* que o constituem (“Operações CIM sobre HTTP” e “Representação de CIM em

XML"), a iniciativa não tem sido acompanhada pelos fabricantes de equipamentos (*hardware* e *software*) pelo que são poucos os que a implementam. Para dificultar ainda mais a uniformização da gestão, alguns fabricantes, como o caso da Microsoft, começaram por implementações proprietárias do WBEM o que, nos próximos anos, irá empecer os esforços até agora desenvolvidos. Apesar deste óbice, a Microsoft tem-se mostrado pioneira nesta área já que os seus sistemas operativos suportam nativamente o modelo de informação comum CIM, possibilitando aos gestores de parques informáticos a capacidade de centralização dos actos de gestão para este tipo de sistemas operativos. Conforme já foi referido nos capítulos anteriores, este aspecto acabou por dominar o caminho percorrido neste trabalho e condicionar a escolha das tecnologias utilizadas na construção da plataforma.

6.2 Perspectivas de Evolução

No que respeita ao trabalho desenvolvido, existe ainda um caminho muito longo a percorrer para que a gestão centralizada de parques informáticos possa ser feita independentemente de qualquer aspecto tecnológico. Foi já referida a dependência da plataforma desenvolvida sobre a tecnologia WMI. Contudo, e tendo em consideração a evolução da plataforma de gestão, esta foi desenvolvida de acordo com uma estrutura modular o que permitirá, em qualquer altura, alterar o método de acesso à informação das máquinas geridas. Mais, dado que o núcleo da plataforma se baseia em *standards* de facto, não proprietários, apenas o módulo de acesso à informação nas máquinas precisa ser alterado. Esta particularidade permitirá que, num futuro recente, quando a iniciativa WBEM for suportada pela grande maioria dos sistemas operativos actuais e for implementada nos mesmos de acordo com as normas definidas, facilmente se faça a migração do método de acesso à informação para um método que respeite as normas e seja universal. Esta é, sem dúvida, a grande perspectiva de evolução da plataforma desenvolvida: a independência da tecnologia implementada na máquina a gerir. A partir do momento em que a plataforma possa gerir uma máquina *Windows* da mesma forma que gere uma máquina *Linux* ou um *MacOS*¸ estará aberto o caminho para a verdadeira gestão centralizada de parques informáticos heterogéneos.

Quanto às funcionalidades da plataforma de gestão, existirão, com certeza, bastantes mais funcionalidades que um administrador possa desejar num sistema deste género. As funcionalidades disponibilizadas poderão ser remodeladas ou outras funcionalidades poderão vir a ser adicionadas de acordo com as necessidades de utilização que poderão surgir num ambiente de produção real. Um outro aspecto que poderá vir a ser alterado no futuro será o sistema de alertas e notificações. Neste momento, trata-se de um sistema independente do modelo CIM logo, não tira partido das reais capacidades do modelo de eventos de CIM. Trata-se portanto de uma funcionalidade que poderá vir a ser melhorada e integrada no

modelo CIM sendo que nesse caso, poderia ser a própria máquina gerida a gerar e a enviar eventos ao sistema gestor.

Nos próximos anos, iremos assistir sem dúvida à crescente implementação destes *standards* pelos vários fabricantes. Começarão a aparecer plataformas cada vez mais completas e com mais funcionalidades para a gestão de qualquer equipamento de qualquer fabricante, independentemente do seu sistema operativo. Por exemplo, para os sistemas operativos *Linux* existe já disponível uma ferramenta que possibilita esta gestão, *OpenPegasus*. Contudo, existem muitos mais sistemas operativos dos quais se espera que venham a aderir a esta iniciativa e assim proliferar a sua utilização.

7 Referências

[1] Yechaim Yemini and German Goldzmidt, “Distributed Management by Delegation”, Proceedings 15th International Conference on Distributed Computing Systems, June 1995

[2] David A. Kelly, “Centralized Management with Decentralized Control”, ebizQ, Março 2004

[3] “The Role of Remote Management in Assuring IT Infrastructure Uptime”, Enterprise Management Associates, Julho 2005

[4] Taisy Silva Webber, “Tolerância a Falhas: Conceitos e Exemplos”, 2001 (http://www.inf.ufrgs.br/~taisy/disciplinas/textos/ConceitosDependabilidade.PDF)

[5] Andrea Westerinen and Winston Bumpus, “The Continuing Evolution of Distributed Systems Management”, Institute of Electronics, Information and Communication Engineers (IEICE) Transactions on Information and Systems, Vol.E86-D NO11 p.2256, Novembro 2003

[6] HP OpenView Management Software, Hewlett-Packard Development Company (http://www.managementsoftware.hp.com/)

[7] CA Unicenter, Computer Associates International Inc., (http://www3.ca.com/solutions/Solution.aspx?ID=315)

[8] J. D. Case, M. Fedor, M. L. Schoffstall, C. Davin, “Simple Network Management Protocol – RFC 1157”, Internet Engineering Task Force (IETF), Maio 1990

[9] J. D. Case, K. McCloghrie, M. Rose, S. Waldbusser, “Introduction to Community-based SNMPv2 – RFC 1901”, IETF, Janeiro 1996

[10] J. Galvin, K. McCloghrie, “Administrative Model for SNMPv2 – RFC 1445”, IETF, Abril 1993

[11] J. Galvin, K. McCloghrie, “Security Protocols for SNMPv2 – RFC 1446”, IETF, Abril 1993

[12] U. Blumenthal, B. Wijnen, “User-based Security Model (USM) for SNMPv3 – RFC 3414”, IETF, Dezembro 2002

[13] B. Wijnen, R. Presuhn, K. McCloghrie, “View-based Access Control Model (VACM) for SNMPv3 – RFC 3415”, IETF, Dezembro 2002

[14] Distributed Management Task Force (http://www.dmtf.org)

[15] Distributed Management Interface (DMI) Specification, DMTF, Janeiro 2003

[16] Common Information Model (CIM) Infrastructure Specification V2.3, DMTF, Outubro 2005

[17] "Common Information Model (CIM) Core Model V2.4", DMTF Whitepaper, Agosto 2000

[18] "The Growing Importance of Management Standards", DMTF Technical Note, Setembro 2003

[19] Alert Standard Format (ASF) Specification V2.0, DMTF, Abril 2003

[20] System Management BIOS Reference Specification V2.4, DMTF, Julho 2004

[21] CIM Core Model V2.5 – LDAP Mapping Specification, DMTF, Abril 2002

[22] J. Hodges, R. Morgan, "Lightweight Directory Access Protocol (LDAP): Technical Specification – RFC 3377", IETF, Setembro 2002

[23] K. Zeilenga, "Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP) – RFC 3383", IETF, Setembro 2002

[24] James Rumbaugh, Michael R. Blaha, William Lorensen, Frederick Eddy, William Premerlani, "Object-Oriented Modeling and Design", Prentice-Hall, Outubro 1990

[25] Grady Booch, James Rumbaugh and Ivar Jacobson, "The Unified Modeling Language: user guide", Addison-Wesley Professional, Setembro 1998

[26] Interface Definition Language, DCE/RPC, The Open group

[27] M. Rose, "A Convention for Defining Traps for Use with the SNMP – RFC 1215", IETF, Março 1991

[28] J. Moy, "Open Shortest Path First (OSPF) v2 – RFC 2328", IETF, Abril 1998

[29] Y. Rekhter and P. Gross, "Application of the Border Gateway Protocol in the Internet", IETF, Março 1995

[30] "Extensible Markup Language (XML)", Version 1.0, W3C Recommendation (http://www.w3.org/TR/REC-xml)

[31] "Specification for the representation of CIM in XML V2.2", DMTF, Dezembro 2004

[32] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.0", IETF RFC 1945, Maio 1996

[33] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", IETF RFC 2068, Janeiro 1997

[34] E. Rescorla and A. Schiffman, “The Secure Hypertext Transfer Protocol”, IETF RFC 2660, Agosto 1999

[35] Craig Tunstall, “The Web and Web Based Enterprise Management” - Technology Whitepaper (http://www.wbem.co.uk)

[36] Elliotte Rusty Harold, “XML 1.1 Bible”, 3rd Edition, Wiley Publishing, Inc., 2004, p.187-308

[37] “XML Schema Part 1: Structures Second Edition”, W3C Recommendation, Outubro 2004 (http://www.w3.org/TR/xmlschema-1/)

[38] “Specification for CIM Operations over HTTP, Version 1.1”, DMTF, Janeiro 2003

[39] H. Nielsen, P. Leach and S. Lawrence, “An HTTP Extension Framework – RFC 2774”, IETF Internet Draft RFC 2774, Fevereiro 2000

[40] T. Berners-Lee, R. Fielding, L. Masinter, “Uniform Resource Identifiers (URI): Generic Syntax”, IETF RFC 2396, Agosto 1998

[41] The Open Group (http://www.opengroup.org)

[42] “OpenPegasus Administrator Guide, Release 2.4”, The Open Group, Abril 2005

[43] Windows Management Instrumentation, Microsoft Corporation

[44] Component Object Model Technologies, Microsoft Corporation (http://msdn.microsoft.com/library/en-us/com/html/9c67dbfd-31ce-4664-a34a-4d26d97e1b1d.asp)

[45] Jeffrey Cooperstein, “Windows Management Instrumentation: Administering Windows and Applications across Your Enterprise”, MSDN Magazine, Maio 2000 (http://msdn.microsoft.com/library/en-us/dnwmi/html/mngwmi.asp)

[46] William Stallings, “SNMP, SNMPv2, SNMPv3, and RMON 1 and 2”, Third Edition, Addison Wesley, 2001

[47] P. Grillo, S. Waldbusser, “Host Resources MIB”, IETF Proposed Standard RFC 1514, Setembro 1993

[48] M. Rose, “Management Information Base for Network Management of TCP/IP-based Internets: MIB-II”, IETF Full Standard RFC 1213, Março 1991

[49] Ethereal, Analisador de protocolos de rede (http://www.ethereal.com)

[50] “Information Systems – Database Language – SQL”, ANSI INCITS 135-1992 (R1998), American National Standards Institute

[51] WMI Query Language, Microsoft Corporation (http://msdn.microsoft.com/library/en-us/wmisdk/wmi/querying_with_wql.asp)

[52] MySQL, MySQL AB Company (http://www.mysql.org)

[53] OpenLdap (http://www.openldap.org)

[54] “Constructing a Moniker String”, Microsoft Corporation

(http://msdn.microsoft.com/library/en-us/wmisdk/wmi/constructing_a_moniker_string.asp)

[55] Lincoln D. Stein, “Network Programming with Perl”, Abril 2001, Addison-Wesley

[56] PRADO, PHP Rapid Application Development Object-Oriented (http://www.xisc.com)

[57] IBM TIVOLI Software (http://www.ibm.com/software/tivoli)

[58] Web-Based Enterprise Management, DMTF (http://www.dmtf.org/standards/wbem)

[59] “XML Document Type Definition v2.1.1”, DMTF, Janeiro 2003

[60] QCODO, PHP Development Framework (http://www.qcodo.com)

[61] PHOCOA, PHP Framework (http://phocoa.com/webapp/public/pages/home)

[62] CakePHP (http://cakephp.org/)

[63] WASP, Web Application Structure for PHP5 (http://wasp.sourceforge.net/content/)

[64] PROPEL, PHP Object Persistence Layer (http://propel.phpdb.org/trac/wiki/WikiStart)

[65] ADOdb, Database Abstraction Library for PHP (http://adodb.sourceforge.net/)

[66] PEAR DB, Database Abstraction Layer (http://pear.php.net/package/DB)

[67] “XML Schema”, W3C Recommendation (http://www.w3.org/TR/xmlschema-0/)

[68] Zend, The php Company (http://www.zend.com)

Anexo A

Open Pegasus

Web-Based Enterprise Management

Neste anexo pretende-se apresentar uma das poucas iniciativas WBEM actuais inteiramente de acordo com as normas e protocolos definidos para o efeito pelo DMTF. Será feita uma breve introdução ao projecto, seguida de uma pequena exposição sobre a arquitectura do mesmo e algumas considerações obre segurança de utilização.

A.1 Introdução

O *Open Pegasus* [42] é uma implementação *open-source* WBEM, desenvolvida de acordo com as normas definidas pelo DMTF, e publicada pelo *Open Group* [41]. O *Open Group* é um consórcio independente de tecnologias e fabricantes, cujo principal objectivo é o desenvolvimento de iniciativas que visam essencialmente o acesso integrado à informação dentro e entre organizações baseado em *standards* de forma a promover a interoperabilidade global entre equipamentos.

Descrito na secção **3.4**, o WBEM é uma iniciativa do DMTF que define quer o modelo para a representação da informação quer o protocolo de comunicação usados para controlar e gerir diversos recursos de um ou mais sistemas. O WBEM define uma especificação para a descrição de dados de gestão e uma especificação do protocolo de comunicação entre entidades para o transporte dos dados de gestão. De acordo com as normas definidas pelo DMTF, o *Open Pegasus* implementa as seguintes especificações:

- Modelo de Informação Comum (secção **3.3**)
- Representação de CIM em XML (secção **3.4.1**)
- Operações CIM sobre HTTP (secção **3.4.2**)

A.2 Arquitectura

A implementação WBEM Open Pegasus consiste num servidor CIM, um conjunto de *providers* e um conjunto de clientes CIM. A interacção entre o servidor CIM e os clientes (aplicações de gestão) é feita através de operações CIM que descrevem uma acção (monitorização/controlo) num recurso gerido pelo servidor, conforme ilustrado na figura 80.

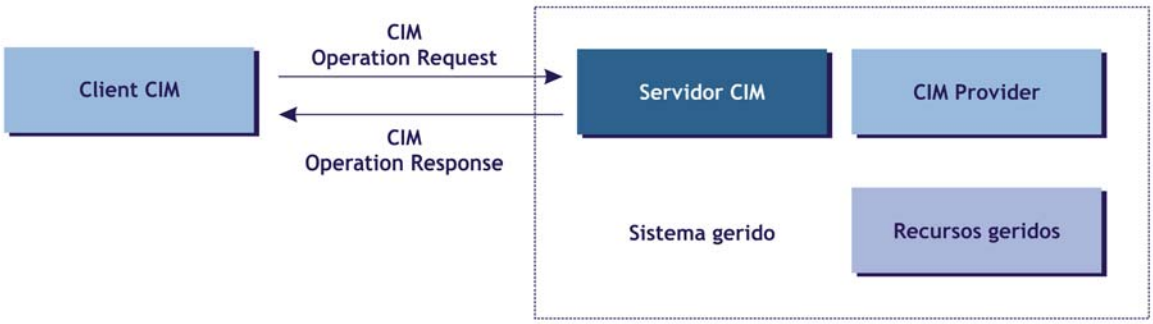


Figura 80 – Componentes da infra-estrutura WBEM *Open Pegasus*

Os clientes CIM enviam pedidos (*Operation Requests*) ao servidor CIM e obtém respostas (*Operation Responses*). O servidor CIM, encarrega-se de processar os pedidos provenientes dos clientes e enviar as respectivas respostas. Internamente, ao nível dos *providers*, é feita a translação entre as operações CIM e as operações de gestão específicas para os recursos geridos. As operações CIM que o servidor *Open Pegasus* é capaz de processar são as suportadas pela norma e que foram descritas na secção **3.4.2**.

A base do *Open Pegasus* assenta, portanto, no servidor CIM. É da responsabilidade deste a recepção dos pedidos dos clientes e respectivo processamento, a manutenção de um repositório de objectos de gestão e a codificação/descodificação das operações CIM.

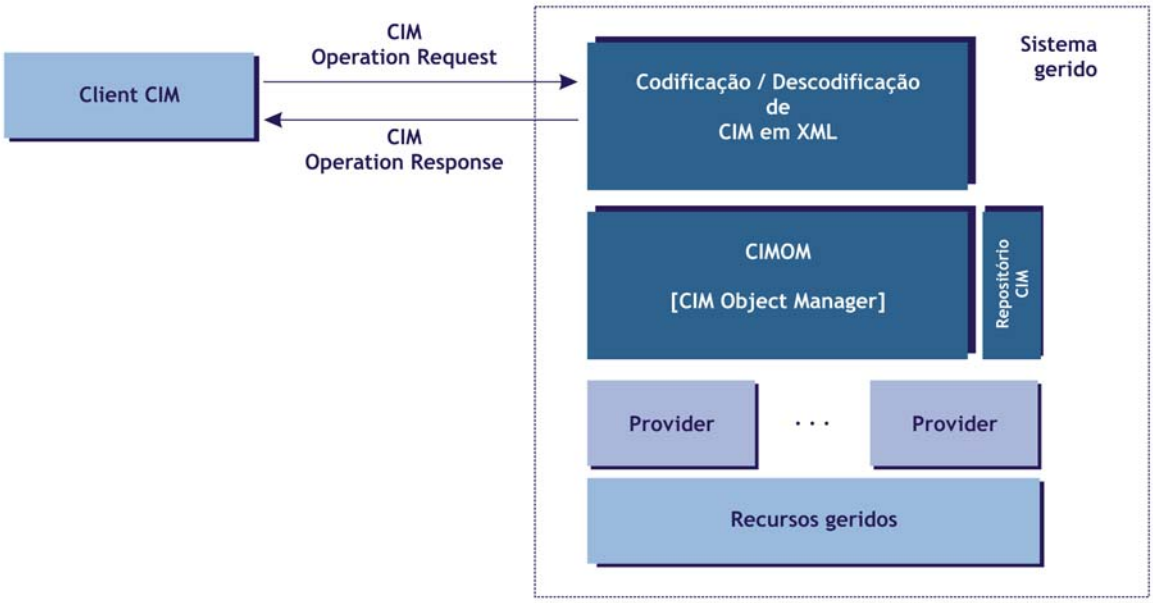


Figura 81 – Arquitectura do servidor CIM *Open Pegasus* (fonte [42])

Na figura 81 apresenta em detalhe a arquitectura do servidor CIM.

A codificação/descodificação de CIM em XML é implementada de acordo com a norma definida para o efeito pelo DMTF. Apesar de o servidor CIM implementar um servidor HTTP, este apenas aceita mensagens CIM válidas, rejeitando todos os outros pedidos. O CIMOM é a entidade responsável por passar os pedidos dos clientes aos *providers* e vice-versa. Além disso, é da sua responsabilidade gerir o repositório de objectos CIM cujo conteúdo são as classes e respectivas descrições relativas aos objectos que podem ser geridos. Note-se que a informação de gestão real é obtida directamente dos *providers* e não de repositório CIM. O repositório apenas indica ao CIMOM a forma como este deverá formatar os dados recebidos dos *providers* e entregá-los ao cliente.

Os *providers* são os componentes que permitem que a infra-estrutura CIM consiga gerir os recursos para os quais são desenvolvidos. Constituem, portanto, a interface entre o CIMOM e os recursos que podem ser geridos. Para que o servidor CIM possa expor a informação de gestão e disponibilizar operações de controlo sobre os recursos aos clientes, é necessário que o *provider* responsável pela interface entre esse recurso e o CIMOM seja registado neste último por forma a tornar o recurso "visível" para a infra-estrutura. Este processo é transparente para o utilizador mas dado que se trata de uma plataforma *open source*, é possível desenvolver e implementar novos *providers* sendo que o registo deles no CIMOM é um passo essencial para a gestão dos recursos para os quais foram desenvolvidos.

A.3 Considerações sobre Segurança

Uma das preocupações que o administrador de sistemas deverá ter em conta é, sem dúvida, a segurança. Dadas as facilidades de gestão remota disponibilizadas pelo *Open Pegasus*, é necessário assegurar que apenas utilizadores autorizados o possam fazer. Neste aspecto, o *Open Pegasus* tem em conta as seguintes situações:

- Pedidos provenientes de utilizadores locais: se o utilizador estiver no mesmo sistema que o *Open Pegasus*, então é considerada válida a autenticação efectuada pelo sistema.
- Pedidos provenientes de utilizadores remotos: o pedido entra via servidor interno HTTP. Neste caso, apenas mensagens válidas são analisadas e cabe ao servidor CIM analisar a informação de utilizador que lhe é passada no cabeçalho da mensagem HTTP (uso de métodos de autenticação HTTP).
- *Providers*: o *Open Pegasus* interage apenas com *providers* registados. Estes, por sua vez, executam no sistema como utilizadores privilegiados.

O *Open Pegasus* implementa ainda um outro nível de segurança, sobre os utilizadores autenticados. É possível, através de uma configuração apropriada do servidor CIM, implementar níveis de autorização para os espaços de nomes ou seja, um utilizador, apesar de autenticado poderá ter apenas privilégios de leitura de um espaço de nomes ou então privilégios de leitura em todos os espaços de nomes e privilégios de escrita apenas em um espaço de nomes.

Anexo B

Troca de informação entre entidades CIM

Análise do payload XML

Este anexo destina-se a apresentar um exemplo de comunicação entre duas entidades CIM. Uma das entidades, designada por cliente, efectua um pedido de enumeração de instâncias para uma classe e a outra entidade, designada por servidor, retorna a resposta ao cliente. A análise da comunicação será feita apenas ao nível do *payload* XML.

B.1 Pedido do Cliente

A tabela seguinte representa o pedido de um cliente CIM para a enumeração das instâncias da classe `PG_OperatingSystem`.

Tabela 5 - Invocação de uma operação CIM em XML

`<?xml version=”1.0” ?>`
`<CIM VERSION=”2.0” DTDVERSION=”2.0”>`
`<MESSAGE ID=”51000” PROTOCOLVERSION=”1.0”>`
	Cabeçalho da mensagem XML, indicando que se trata de uma mensagem CIM

`<SIMPLEREQ>` `<IMETHODCALL NAME=”EnumerateInstances”>`	Pedido simples para execução da operação: *EnumerateInstances*

`<LOCALNAMESPACEPATH>`
`<NAMESPACE NAME=”root” />`
`<NAMESPACE NAME=”cimv2” />`
`</LOCALNAMESPACEPATH>`
	Especificação do espaço de nomes sobre o qual será feita a operação CIM.

`<IPARAMVALUE NAME=”ClassName”>`
`<CLASSNAME NAME=”PG_OperatingSystem” />`
`</IPARAMVALUE>`
	Classe sobre a qual é pedida a enumeração de instâncias: `PG_OperatingSystem`

`</IMETHODCALL>` `</SIMPLEREQ>`	Fecho da invocação do método (operação) e do pedido
`</MESSAGE>` `</CIM>`	Fecho da mensagem CIM

B.2 Resposta do Servidor

Após receber a mensagem de operação CIM, o servidor processa o conteúdo XML, passando a invocação do método ao *provider* correspondente à classe `PG_OperatingSystem`, que se encarregará de aceder ao(s) recurso(s) em causa e devolverá a resposta ao CIMOM que por sua vez irá construir uma nova mensagem XML com a resposta. Parte dessa mensagem é apresentada na tabela seguinte.

Tabela 6 - Resposta a uma operação CIM em XML

`<?xml version=”1.0” encoding=”utf-8”?>`
`<CIM VERSION=”2.0” DTDVERSION=”2.0”>`
`<MESSAGE ID=”51000” PROTOCOLVERSION=”1.0”>`
	Cabeçalho da mensagem XML, indicando que se trata de uma mensagem CIM

`<SIMPLERSP>`
`<IMETHODRESPONSE`
`NAME=”EnumerateInstances”>`
	Resposta ao pedido simples de enumeração de instâncias

`<IRETURNVALUE>` `<VALUE.NAMEDINSTANCE>`	Valor de retorno são todas as instâncias da classe
`<INSTANCENAME` `CLASSNAME=”PG_OperatingSystem”>`	Início das propriedades chave da classe

`<KEYBINDING NAME=”CreationClassName”>`
`<KEYVALUE VALUTYPE=”string”>`
`CIM_OperatingSystem`
`</KEYVALUE>`
`</KEYBINDING>`
	Uma das chaves é a propriedade `CreationClassName` que indica a classe ou subclasse a partir da qual foi criada a instância

`<KEYBINDING`
`NAME=”CSCreationClassName”>`
`<KEYVALUE VALUETYPE=”string”>`
`CIM_UnitaryComputerSystem`
`</KEYVALUE>`
`</KEYBINDING>`
	Outra propriedade chave é a `CSCreationClassName`. Indica a classe a partir da qual foi instanciada a classe correspondente ao Sistema: `CIM_UnitaryComputerSystem`

`<KEYBINDING NAME=”CSName”>`
`<KEYVALUE VALUETYPE=”string”>`
`bilbo`
`</KEYVALUE>`
`</KEYBINDING>`
	Outra propriedade chave é o nome do Sistema. `CSName` vem de *Computer System Name*

`<KEYBINDING NAME=”name”>`
`<KEYVALUE VALUETYPE=”string”>`
`Fedora Core`
`</KEYVALUE>`
`</KEYBINDING>`
	Nome do sistema operativo do Sistema.

`</INSTANCENAME>`	Fim das propriedades chave da instância
`<INSTANCE` `CLASSNAME=”PG_OperatingSystem”>`	Início da enumeração das propriedades da instância

`<PROPERTY`
`NAME=”CSCreationClassName”`
`TYPE=”string”>`
	Novamente a listagem das propriedades chave da instância

```
        <VALUE>CIM_UnitaryComputerSystem
        </VALUE>
       </PROPERTY>
       <PROPERTY NAME=”CSName”
          TYPE=”string”>
        <VALUE>bilbo</VALUE>
       </PROPERTY>
       <PROPERTY
          NAME=”CreationClassName”
          TYPE=”string”>
        <VALUE>CIM_OperatingSystem
        </VALUE>
       </PROPERTY>
       <PROPERTY NAME=”Name”
          TYPE=”string”>
        <VALUE>Fedora Core</VALUE>
       </PROPERTY>
       <PROPERTY NAME=”Caption”
          TYPE=”string”>
        <VALUE>The current Operating System</VALUE>
       </PROPERTY>
       <PROPERTY NAME=”OSType”
          TYPE=”string”>
        <VALUE>Linux (36)</VALUE>
       </PROPERTY>

                   ...

       <PROPERTY
          NAME=”OperatingSystemCapability”
          TYPE=”string”>
        <VALUE>32 bit</VALUE>
       </PROPERTY>
      </INSTANCE>
     </VALUE.NAMEDINSTANCE>
    </IRETURNVALUE>
   </IMETHODRESPONSE>
  </SIMPLERSP>
 </MESSAGE>
</CIM
```

Listagem das restantes propriedades da instância

Fecho de instância e final de mensagem

Anexo C

Monitorização/Gestão via WMI com PHP

Casos Práticos

Neste anexo são apresentados alguns casos práticas de gestão WMI utilizando PHP. Dada a imensa variedade de informação e de actos de gestão que podem ser realizados, são colocados 3 casos, pretendendo cobrir as principais operações que são realizadas pela plataforma de gestão. A primeira trata-se de uma consulta simples e obtenção das instâncias de uma classe. No segundo caso, é colocado um exemplo do modo como podem ser tratadas as associações do modelo de informação comum. Por fim, é apresentado um exemplo de actuação na máquina gerida, recorrendo à invocação de métodos sobre instâncias de classes.

Os exemplos apresentados não contêm o código relativo à fase de estabelecimento de ligação com a máquina alvo visto já ter sido abordado em **4.4.2.1**.

C.1 Listagem de Processos

O código apresentado a seguir permite obter a listagem dos processos em execução na máquina consultada. Trata-se de uma consulta simples, onde são processadas as várias instâncias da classe correspondente aos processos.

```
/*
 * Construção da query e execução da mesma sobre a ligação estabelecida,
 * representada pelo objecto $wbem. A variável $objects conterá as
 * instâncias retornadas.
*/
$query = "SELECT * FROM WIN32_Process";
$objects = $wbem->execquery($query);

/*
 * Inicialização do array associativo que conterá a informação acerca
 * dos processos
*/
```

```
$processes = array();

/*
 * Ciclo para preenchimento de um array associativo com a informação
 * relativa aos processos. Para simplificar são apresentados apenas
 * algumas propriedades dos processos. Cada iteração do ciclo corresponde
 * a uma instância da classe Win32_Process ou seja, um processo.
*/
foreach($objects as $object) {

  $processes[] = array(
    /* Identificador (handle) do processo */
    'handle' => $object->handle,
    /* Pequena descrição textual do processo */
    'processo' => $object->caption,
    /* Data/hora em que o processo foi lançado */
    'initDate' => $object->creationdate,
    /* Path de execução do processo */
    'path' => $object->executablepath,
    /* Memória gasta pelo processo (valor instantâneo) */
    'memory' => round($object->workingsetsize/1000,0),
  );

}
```

C.2 Interdependência entre Serviços

Muitos serviços num sistema operativo Windows são dependentes entre si. Enquanto alguns requerem um determinado estado por parte de outros serviços, outros há em que sobre os quais dependem vários serviços. O código que se apresenta a seguir exemplifica o tipo de *query* que deve ser usada, tendo em conta as associações entre os vários serviços e que determinam as suas dependências.

```
/*
 * Inicialização dos arrays associativos que conterão:
 * - os serviços sobre os quais está dependente o serviço $service
 * - os serviços que dependem do serviço $service
*/
$dependOnServices = array();
$dependentOfServices = array();

/*
 * Query de associação para obtenção dos serviços associados sobre a
 * forma: serviços sobre os quais $service está dependente
*/
```

```
   $assocQuery = "ASSOCIATORS OF {WIN32_Service.Name='$service'} WHERE
AssocClass=WIN32_DependentService Role=Dependent";
   $dependOn = $wbem->execquery($assocQuery);

   /*
    * Ciclo para preenchimento de um array associativo com o nome e o
    * estado dos serviços
   */
   foreach($dependOn as $service) {
     $dependOnServices[] = array(
        'service' => $service->name,
        'state' => $service->state
     );
   }

   /*
    * Query de associação para obtenção dos serviços associados sobre a
    * forma: dependentes do serviço $service
   */
   $assocQuery = "ASSOCIATORS OF {WIN32_Service.Name='$name'} WHERE
AssocClass=WIN32_DependentService Role=Antecedent";
   $dependentOf = $wbem->execquery($assocQuery);

   /*
    * Ciclo para preenchimento de um array associativo com o nome e o
    * estado dos serviços
   */
   foreach($dependentOf as $service) {
     $dependentOfServices[] = array(
        'service' => $service->name,
        'state' => $service->state
     );
   }
```

C.3 Execução remota de métodos

Além da consulta de dados, é também possível, via WMI, executar operações ou alterar configurações remotamente. Para tal, é necessário executar métodos sobre as instâncias das classes em que se quer actuar. Existem dois métodos que podem ser usados para a execução de métodos. O primeiro, *execquery* (já visto nos exemplos anteriores), corresponde à invocação do método directamente sobre o *provider* WMI, designado por acesso directo. No entanto, em alguns métodos, este acesso directo não pode ser usado. O que acontece é que alguns métodos requerem parâmetros de *output* o que não é suportado pelo PHP logo, não é possível assignar o(s) valore(s) de retorno a

variáveis passadas como parâmetros. Como forma de rodear esta situação, pode ser usado um segundo método, *execmethod*, onde o(s) valore(s) de retorno são devolvidos como atributos de um objecto resultado.

De seguida é apresentado um exemplo para cada um dos tipos: a renovação remota de uma licença de DHCP; obtenção do *owner* de um processo em execução.

```
   /*
    * Query de associação entre uma instância da classe Win32_NetworkAdapter
    * (identificada pelo DeviceID $ifindex) e a correspondente instância da
    * classe de configuração do adaptador (Win32_NetworkAdapterSetting).
   */
   $assocQuery = "ASSOCIATORS OF {WIN32_NetworkAdapter.DeviceID='$ifindex'}
WHERE AssocClass=WIN32_NetworkAdapterSetting";
   $objects = $wbem->execquery($assocQuery);

   /*
    * Mesmo tratando-se de uma única instância, o resultado da invocação da
    * query consiste numa lista de objectos, neste caso com apenas uma
    * instância pelo que é usado um ciclo.
   */
   foreach($objects as $object) {
      /*
       * A renovação da licença de DHCP é feita invocando o respectivo
       * método sobre a instância da classe.
      */
      $retCode = $object->RenewDHCPLease();
   }
```

```
   /*
    * Execução do método GetOwner sobre a instância da classe Win32_Process
    * identificado pelo Handle $handle.
   */
   $outParams = $wbem->execmethod(
      'Win32_Process.Handle="'.$handle.'"',
      'GetOwner'
   );

   /*
    * A variável $outParams contém os dois atributos de retorno.
   */
   $owner['user'] = $outParams->user;
   $owner['domain'] = $outParams->domain;
```

Anexo D

Gestão de Tarefas Automáticas em Perl

No seguimento do que foi referido em **4.4.2.2**, a gestão de tarefas automáticas e respectiva integração com o sistema operativo foi feita recorrendo a um módulo específico do Perl (`Win32::TaskScheduler`) que integra directamente com o sistema operativo. Além das tarefas que executam apenas uma vez (código já apresentado na Figura 33) , neste anexo é apresentado o código para a criação de tarefas automáticas com as várias periodicidades disponibilizadas pela plataforma: execução por hora, execução por dia, execução por semana e execução por mês.

D.1 Execução por Hora

Este tipo de tarefas tem uma periodicidade horária. A sua configuração inclui aspectos como: hora de início, hora de fim, taxa de repetição por hora, dias da semana em que executa..

```
/*
 * Inclusão do módulo para gestão de tarefas
*/
use Win32::TaskScheduler;

/*
 * Inicialização do objecto COM que tornará possível o acesso à
 * Scheduler API do sistema.
*/
$sched = Win32::TaskScheduler->New();

/*
 * Configuração temporal da tarefa. É definida uma data e hora para a
 * activação da mesma, além da definição do tipo de tarefa.
 * Apesar de se tratar de uma tarefa com periodicidade horária, foi usado
 * o tipo TASK_TIME_TRIGGER_WEEKLY dado que permite a configuração
 * adicional dos dias em que a tarefa será executada.
```

```
*/
%trigger = (
    'StartHour' => $hour,
    'StartMinute' => $minute,
    'BeginDay' => $day,
    'BeginMonth' => $month,
    'BeginYear' => $year,
);
$trigger{'TriggerType'} = $sched->TASK_TIME_TRIGGER_WEEKLY;

/*
 * Restante configuração da tarefa. Neste caso ela irá executar de
 * segunda a sexta, todas as semanas.
*/
$trigger{'Type'} = {
     'DaysOfTheWeek' => $sched->TASK_MONDAY
             | $sched->TASK_TUESDAY
             | $sched->TASK_WEDNESDAY
             | $sched->TASK_THURSDAY
             | $sched->TASK_FRIDAY
     'WeeksInterval' => 1,
};

/*
 * Além dos dias da semana em que a tarefa é executada, é ainda definida
 * a duração diária da tarefa. O valor $duration indica o número de
 * minutos em que, durante o dia, a tarefa estará activa. Por exemplo, se
 * a hora de início for 10h00 e a duração 120 minutos, então a tarefa
 * será executada entre as 10h00 e as 12h00, o número de vezes definido
 * pela taxa de repetição $repeatInterval. Caso $repeatInterval seja 2, a
 * tarefa será executada duas vezes.
*/
$trigger{'MinutesDuration'} = $duration;
$trigger{'MinutesInterval'} = $repeatInterval;

/*
 * Criação da tarefa com o nome $taskName, alocando-lhe o trigger
 * configurado.
*/
$sched->NewWorkItem($taskName,\%trigger);

/*
 * Aplicação que será lançada pelo scheduled job e respectivo directório
 * de trabalho.
*/
$sched->SetApplicationName($app);
$sched->SetWorkingDirectory($workDir);
```

```
/*
 * Para finalizar, o scheduled job configurado é gravado no sistema
 * operativo, sendo nesta altura libertado o objecto $sched.
*/
$sched->Save();
```

D.2 Execução por Dia

As tarefas de periodicidade diária são tarefas que executam uma vez por dia, a uma determinada hora, nos dias da semana configurados pelo utilizador.

```
/*
 * Inclusão do módulo para gestão de tarefas
*/
use Win32::TaskScheduler;

/*
 * Inicialização do objecto COM que tornará possível o acesso à
 * Scheduler API do sistema.
*/
$sched = Win32::TaskScheduler->New();

/*
 * Configuração temporal da tarefa. É definida uma data e hora para a
 * activação da mesma, além da definição do tipo de tarefa.
 * Apesar de se tratar de uma tarefa diária, foi usado o tipo
 * TASK_TIME_TRIGGER_WEEKLY dado que permite a configuração adicional
 * dos dias em que a tarefa será executada.
*/
%trigger = (
    'StartHour' => $hour,
    'StartMinute' => $minute,
    'BeginDay' => $day,
    'BeginMonth' => $month,
    'BeginYear' => $year,
);
$trigger{'TriggerType'} = $sched->TASK_TIME_TRIGGER_WEEKLY;

/*
 * Restante configuração da tarefa. Neste caso ela irá executar de
 * segunda a sexta, todas as semanas.
*/
$trigger{'Type'} = {
     'DaysOfTheWeek' => $sched->TASK_MONDAY
```

```
                | $sched->TASK_TUESDAY
                | $sched->TASK_WEDNESDAY
                | $sched->TASK_THURSDAY
                | $sched->TASK_FRIDAY,
        'WeeksInterval' => 1,
};

/*
 * Criação da tarefa com o nome $taskName, alocando-lhe o trigger
 * configurado.
*/
$sched->NewWorkItem($taskName,\%trigger);

/*
 * Aplicação que será lançada pelo scheduled job e respectivo directório
 * de trabalho.
*/
$sched->SetApplicationName($app);
$sched->SetWorkingDirectory($workDir);

/*
 * Para finalizar, o scheduled job configurado é gravado no sistema
 * operativo, sendo nesta altura libertado o objecto $sched.
*/
$sched->Save();
```

D.3 Execução por Semana

A periodicidade semanal é utilizada em tarefas que necessitam de ser executadas apenas uma vez por semana. O exemplo que se segue apresenta a criação de uma tarefa que será executada todas as segundas.

```
/*
 * Inclusão do módulo para gestão de tarefas
*/
use Win32::TaskScheduler;

/*
 * Inicialização do objecto COM que tornará possível o acesso à
 * Scheduler API do sistema.
*/
$sched = Win32::TaskScheduler->New();

/*
 * Configuração temporal da tarefa. É definida uma data e hora para a
```

```
 * activação da mesma, além da definição do tipo de tarefa.
*/
%trigger = (
    'StartHour' => $hour,
    'StartMinute' => $minute,
    'BeginDay' => $day,
    'BeginMonth' => $month,
    'BeginYear' => $year,
);
$trigger{'TriggerType'} = $sched->TASK_TIME_TRIGGER_WEEKLY;

/*
 * Restante configuração da tarefa. Será executada às segundas, todas
 * as semanas.
*/
$trigger{'Type'} = {
     'DaysOfTheWeek' => $sched->TASK_MONDAY,
     'WeeksInterval' => 1,
};

/*
 * Criação da tarefa com o nome $taskName, alocando-lhe o trigger
 * configurado.
*/
$sched->NewWorkItem($taskName,\%trigger);

/*
 * Aplicação que será lançada pelo scheduled job e respectivo directório
 * de trabalho.
*/
$sched->SetApplicationName($app);
$sched->SetWorkingDirectory($workDir);

/*
 * Para finalizar, o scheduled job configurado é gravado no sistema
 * operativo, sendo nesta altura libertado o objecto $sched.
*/
$sched->Save();
```

D.4 Execução por Mês

Além da hora e dos meses em que são executadas, as tarefas de periodicidade mensal podem ser definidas de duas maneiras: ou o utilizador define o dia do mês em que pretende

que ela seja executada ou então define o tipo de dia que pretende (ex. 1ª segunda, última sexta, etc.).

```
/*
 * Inclusão do módulo para gestão de tarefas
*/
use Win32::TaskScheduler;

/*
 * Inicialização do objecto COM que tornará possível o acesso à
 * Scheduler API do sistema.
*/
$sched = Win32::TaskScheduler->New();

/*
 * Configuração temporal da tarefa. É definida uma data e hora para a
 * activação da mesma.
*/
%trigger = (
    'StartHour' => $hour,
    'StartMinute' => $minute,
    'BeginDay' => $day,
    'BeginMonth' => $month,
    'BeginYear' => $year,
);

/*
 * Existem 2 maneiras de definir a periodicidade. Ou se indica o dia
 * do mês em que é executada ou define-se o tipo de dia.
*/
if($day) {

    /*
     * Se for definido o dia de execução (1..31), é definido um trigger
     * do tipo TASK_TIME_TRIGGER_MONTHLYDATE ao qual é adicionado os
     * meses em que a tarefa será executada.
    */
    $trigger{'TriggerType'} = $sched->TASK_TIME_TRIGGER_MONTHLYDATE;
    $trigger{'Type'} = {
         'Months' => $sched->TASK_JANUARY
               | $sched->TASK_MARCH
               | $sched->TASK_MAY
               | $sched->TASK_JULY
               | $sched->TASK_SEPTEMBER
               | $sched->TASK_NOVEMBER,
    };
} else {
```

```
    /*
     * Caso contrário, o trigger é do tipo TASK_TIME_TRIGGER_MONTHLYDOW
     * o que significa que é necessário adicionar, além dos meses, qual a
     * semana (1ª, 2ª …) e em que dia se pretende executar a tarefa. Neste
     * exemplo, a tarefa será executada na primeira segunda dos meses
     * seleccionados.
    */
    $trigger{'TriggerType'} = $sched->TASK_TIME_TRIGGER_MONTHLYDOW;
    $trigger{'Type'} = {
         'WhichWeek' => $sched->TASK_FIRST_WEEK,
         'DaysOfTheWeek' => $sched->TASK_MONDAY,
         'Months' => $sched->TASK_JANUARY
              | $sched->TASK_MARCH
              | $sched->TASK_MAY
              | $sched->TASK_JULY
              | $sched->TASK_SEPTEMBER
              | $sched->TASK_NOVEMBER,
    };
}

/*
 * Criação da tarefa com o nome $taskName, alocando-lhe o trigger
 * configurado.
*/
$sched->NewWorkItem($taskName,\%trigger);

/*
 * Aplicação que será lançada pelo scheduled job e respectivo directório
 * de trabalho.
*/
$sched->SetApplicationName($app);
$sched->SetWorkingDirectory($workDir);

/*
 * Para finalizar, o scheduled job configurado é gravado no sistema
 * operativo, sendo nesta altura libertado o objecto $sched.
*/
$sched->Save();
```